剛体粒子による流れの統計力学的シミュレーション(修士論文(2001年度))

AUTHOR(S):

石渡, 竜也

CITATION:

石渡, 竜也. 剛体粒子による流れの統計力学的シミュレーション(修士論文(2001年度)). 物性研究 2002, 78(6): 717-751

ISSUE DATE:

2002-09-20

URL:

http://hdl.handle.net/2433/97280

修士論文（2001年度）

剛体粒子による流れの統計力学的シミュレーション

東京大学大学院工学系研究科物理工学専攻
石渡竜也

目 次

第1章　目的

本研究の目的は完全剛体粒子を用いたEvent-Driven型分子動力学(Molecular Dynamics:MD)シミュレーション[1]を用いて流体のシミュレーションを行い、移動境界問題への有用性を調べることである。非圧縮粘性流のシミュレーションは連続体近似に基づいた差分法で行われることが多く、流体を粒子の集まりとして取り扱う方法は少ない。しかし連続体近似に基づく方法では移動する境界条件に対応することが原理的に困難である。流体を粒子として、しかも剛体粒子として扱うことができれば、空間の離散化に伴う数学的に複雑な操作や計算中の数値発散の危険などを比較的容易に避けることができると考えられる。さらに移動する境界条件にも容易に取り扱うことが可能である。

分子動力学法は大量の粒子を扱うため、計算量が非常に大きくなり効率が悪いとして過去の研究ではあまりよい方法と考えられていなかったようであるが、近年の計算能力の向上によってある程度の規模の問題が解けるようになってきている。今後計算機自体の高速化やアルゴリズムの改良によって計算能力がさらに向上すれば、さらに大規模な問題が解けるようになるであろう。そうなれば相対的に計算負荷は小さな問題になり、複雑な数学的処理を考える必要がなく、また安定性などを気にする必要がないメリットは大きいと確信する。

以上の目的のため、本研究ではまずEvent-Driven型分子動力学シミュレーションで流体のシミュレーションを行うための基礎データとして、粘性率の系の諸定数への依存性を調べた。次に円柱を過ぎる流れに分子動力学法を適用し、実験事実と合う計算結果を得られることを示す。最後に、界面を持つ流れや運動物体を含む流れなど、分子動力学法の長所が生きると考えられる問題に応用した結果を示す。

1.1　分子動力学法による流体のシミュレーション

分子動力学法によって流体を再現した例としては、Rapaportによって行われたソフトコアの分子動力学法によるカルマン渦の再現[1]が印象的である。これは粒子間に引力のないポテンシャルを仮定した二次元MDシミュレーションであり、円柱後方の渦放出の様子を再現したものである。シミュレーションに用いた粒子数は17万粒子と年代を考えると非常に大きな計算を行っており、粒子の面積あたり密度は0.83で熱速度は0.2に設定、粒子に重力のような力を与えることで流体を駆動して流れを生み出し後流を観測している。この論文では円柱後方のカルマン渦の発生が再現されていて、発生過程を時間を追って表示している。この論文の残念なところは、レイノルズ数別の流れパターンの変化が示されておらず、すべて後方にカルマン渦が発生している結果のみが示されている。また概算されたレイノルズ数からは円柱後方に付着渦が発生すべき計算条件にある場合であってもカルマン渦が発生しているなど不自然な点もあった。

その後Rapaportは完全剛体球でのRayleigh-Bénard対流の再現[2]を試みた。この研究は流れパターンが再現されたことに重点が置かれており、差分法による流体解析との比較などは行われていない。その後のPuhlの論文[3]ではRayleigh-Bénard対流について差分法による流体解析との比較が行われており、境界の影響が強く残っているためあまりよく合わなかったという結果が得られている。

[1]分子動力学と言ってもこの場合、粒子一つ一つが分子を表現しているわけではない。単に流体の表現として粒子を用いているものである。言葉としては粒子動力学と表記したほうが適当かもしれないが、Rapaportなどが Molecular Dynamicsという名前で呼んでいるため、それにあわせて分子動力学とした。

以上のように、これまでの研究ではレイノルズ数に依存した流れパターンの変化を再現していない。また抗力やストローハル数など、流体のシミュレーションが行われる際に計算精度を見るために実験値と比較するような値が測定されていない。このように、これまでの研究は決して十分なものとは言えず、剛体粒子 MD を粘性流体の再現に用いるための基礎となる資料は不足している。こうした現状に鑑み、本研究では剛体粒子系の粘性率測定という最も基本的な部分の検証から始めることにする。粘性率は流体の性質を記述する最も基本的な物性定数の一つである。非圧縮粘性流れにおいては流れパターンがレイノルズ数に依存するが、粘性率はレイノルズ数を計算するために不可欠な物性定数である。その後、測定された粘性率を利用して円柱周りの流れを流れパターン別に再現し、連続体近似に基づくシミュレーションとどれだけ一致するかを調べることにする。

1.2 格子を使った移動境界のシミュレーション

境界値問題とは、ある微分方程式について境界で与えられた条件を満たす解を見つける問題である。移動境界問題は、その中でも境界条件が時間的に変化するものを指す。流体の運動を解く際には、解くべき微分方程式はナビエストークス方程式である。流体で言う移動境界問題とは例えば砂粒や砂利のたくさん混じった水の流れである。このような移動境界の問題を計算し始めた動機は砂の輸送などを砂粒子のレベルから計算したいと考えたからである。砂丘の運動や水底の砂の様子などを計算したシミュレーションでは、堆積した砂の形状を境界条件として流体部分のシミュレーションを行い、得られた速度場からモデル方程式を用いて砂の輸送量を計算して形状の変化とするやり方が採用されている [4][5]。ここで砂の混じった流体を直接計算することができればモデル方程式の検証計算を行うことができ、またはモデルなしで直接砂丘などのシミュレーションを行うことができる。

砂粒や砂利の混じったものを扱う場合、水は流体として扱ってよいが砂粒や砂利は剛体として扱わなければならない。流体の立場から見れば砂利表面は境界条件であり、空間に多数の刻々と変化する境界条件の設定された流れを解くことになる。こういった問題は通常用いられる格子を用いる方法では取り扱いが困難である。

格子を用いる流体シミュレーション方法の代表的な例は MAC 法である。MAC 法は非圧縮粘性流体の解法としてよく用いられる方法であり、多くの解法が MAC 法から派生して開発されている。以下では MAC 法に準じて移動境界を含む問題を取り扱う際の困難を見ていこう。MAC 法では非圧縮で無次元化されたナビエストークス方程式と連続の式

$$\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v}\cdot\nabla)\mathbf{v} = -\nabla p + \frac{1}{\mathrm{Re}}\nabla^2\mathbf{v} \tag{1.1}$$

$$\nabla\cdot\mathbf{v} = 0 \tag{1.2}$$

(ρ は密度、ν は動粘性率)

を基礎方程式として用い、差分化を行う。簡単のため系は二次元であるとし、変数は流速の x 成分を u、y 成分を v として図 1.1 のように互い違いに配置する。変数は格子を組んでいるので差分によって xy 方向の微分を構成することは容易であり、また uv が正方形領域から流入、流出する流体の量を表現しているため連続の式を満たすことが容易である。

ここで境界条件が矩形の領域で与えられているとする、例えば計算領域が長方形の領域であったり、障害物が格子と同じ向きの正方形の形をしている場合などである。境界条件は多くの場合流速で与えられるが、矩形の場合は u や v の計算点がちょうど境界上に乗るため境界での u や v の値を所定の値に固定すればよく、さしたる困難は伴わない。

この方法で円盤や球の表面のような曲面に対応することを試みたとしよう。状況は図 1.2 のようになる。このままでは計算点が境界上に乗らず曲面境界を正しく表現することができない。境界上でのみ計算点をずらして対応することもできるが、格子の長さが周囲と大きく異なるようになってしまうと計算精度が低下する。そこで曲面に対応するためには座標変換を行い、図 1.3 のような曲面に沿った格子を用いて計算を行うのが一般的である。このような座標系を境界適合座標系と呼ぶ。例えば円柱周りの流れであれば 2 次元極座標のような形に座標変換をし、物体が円柱でなく他の形状の場合にはその形状に合った形に再変換すればよい。変換は解析的である必要はなく、数値的でよい。計算に用いる微分関係式の座標変換は数値的な変換関係を用いて変換することができる。境界で流入速度などの条件を課す都合上、境界上で格子は垂直になっていることが望ましい。また格子の間隔が隣の格子と大きく異なると精度が下がるので格子間隔を変化させるときは緩やかになるように変換する。

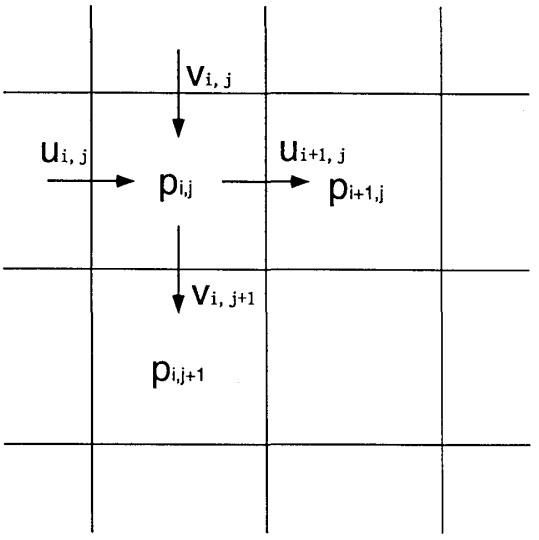

図 1.1: MAC 法における変数の配置

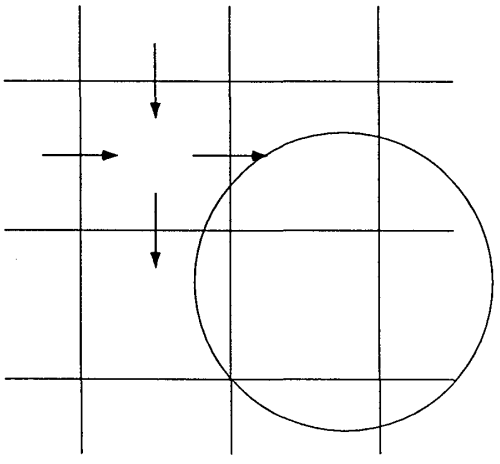

図 1.2: MAC 法における曲面に対する変数の配置

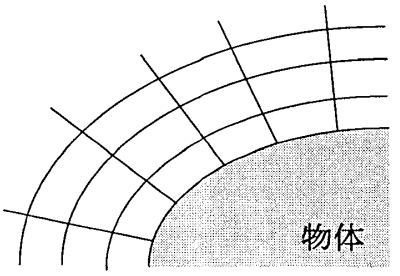

図 1.3: 曲面に対応する格子

一つの物体を過ぎる流れのような場合には以上のような方法で十分に計算が可能である。しかしこれが運動する複数の物体になると不都合が出てくる。一つの物体の場合は、極座標ないしその変形のような形に変換すればよかったが、しかし物体が複数になるとこの座標変換は容易ではない。しかも運動する物体が対象であれば、物体の位置は各時間ステップごとに変化してしまう。これに対応するためには各時間ステップごとに座標変換をやり直さなければならず、そのための計算コストは非常に大きくなってしまう。このため境界適合座標を用いて運動する多数の物体を含む流れを計算することは困難である。

この困難を避ける方法の一つとして、座標変換を止めて直交座標系で解こうという方法が考えられる。本論文の研究を始める以前に MAC 系の解法の一種である MAC 法で 2 物体の運動を計算したことがある。設定状況としては以下のような状況である。2 次元で左から右へと流体の流れ

る流路内に2つの円形物体を置き、流れが成長するまで物体を固定しておく。ある程度時間が経過し物体後方の渦が成長したころを見計らって物体の固定を解除すると、物体は流されて右に流れ始め、互いの相互作用によって複雑な軌跡を描くというものである。図1.5がその途中経過の図であり、図1.4が2粒子の軌道のプロットである。

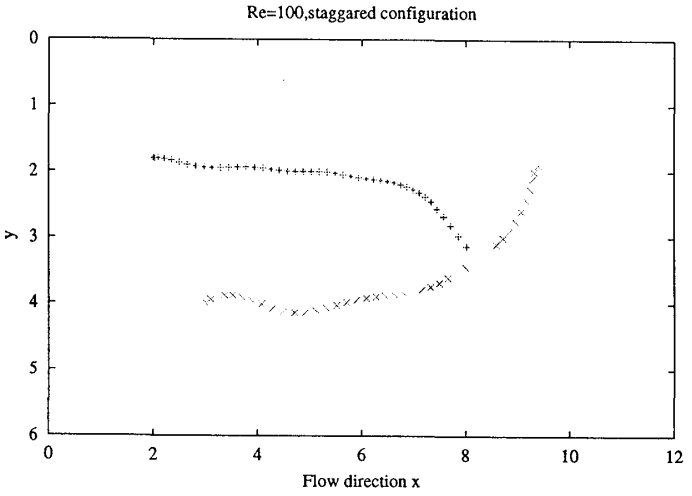

図 1.4: 2粒子の軌道。物体が運動を開始する$t = 2$から0.5おきにプロットしたもの。流れは左から右。プロットの左端が最初の位置。

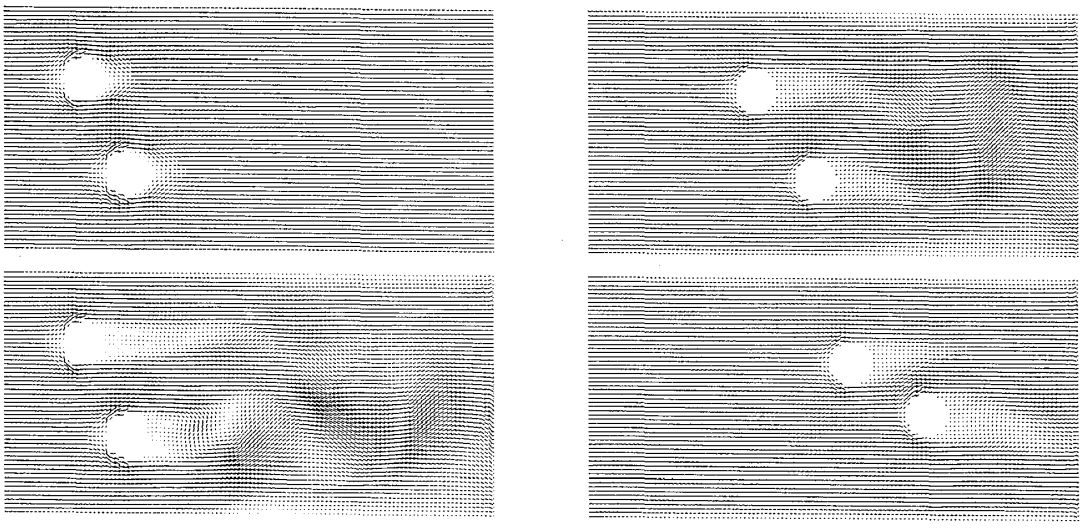

図 1.5: 2粒子の軌道計算における瞬時流れ場。線は流速を示している。線が濃くなっている所は流れの速い場所であり、白く抜けている所は渦を巻いているところである。2物体は$t = 0$から$t = 2$まで固定されており、$t = 2$に固定が解除され運動を開始する。図は左上、左下、右上、右下の順にt=0.05 , 2 , 2.75 , 3.5

しかしこの方法で多数の物体を含む流れを計算する事は困難であった。まず境界がステップ状に扱われて近似精度が低下しているため、静止円柱に対する計算を行った場合に抗力の値が実験値から大きくずれる (2倍程度) という不都合が見られた。それだけであれば運動粒子の計算に用いても定性的には正しい結果をもたらすと考えられるが、運動粒子の計算をしていると計算が破

綻することが多い。物体が多くなると必然的に物体間の衝突が多発するが、衝突前後で計算が不安定になることが多いのである。原因としては粒子が衝突する寸前に粒子の隙間にある流体を外に押しやる速度が大きく不安定性を増大させることや、衝突によって連続的であった流体が分断されることによる影響などが考えられる。

このような理由により衝突によって計算が不安定になるようでは多数の砂粒がぶつかり合う砂山の計算などには適用できなかった。そもそも格子は空間に固定されており時間とともに大きく変形することに対応するようにできていない、この点で格子を用いる方法は多数の運動物体を含む流れの再現に向いていないといえるだろう。

今回 MD で粒子による流体の再現を試みたのはこのような格子の処理に伴う困難を回避できるためである。粒子が流体を表現しているため格子を作成する必要がない。これにより複数の物体に適合する格子を生成する手間と座標変換の煩雑さから解放される。

さらに利点として挙げられるのはその安定性である。差分近似の精度は一般にテーラー展開で評価され、「n 次の精度」という形で表現される。これは逆にいうと $n+1$ 次の誤差は許容されているわけであり、その誤差によってエネルギーや運動量が保存されず数値発散を引き起こす事がある。それに対し Event-Driven 型 MD では衝突処理が瞬間的に行われるように近似するため、エネルギー保存と運動量保存は厳密に維持される。そのため数値発散の危険はない。

第2章　計算方法

本研究で計算モデルとして用いたのは完全剛体円盤であり、Event-Driven 型のアルゴリズムで計算を行っている。剛体以外の相互作用ポテンシャルを持つ粒子の場合、一定の時間刻みに従って時間発展を求めていく方法 (Time-Step-Driven Molecular Dynamics:TSDMD) が用いられる。しかし剛体の場合には、粒子系が衝突時にのみ相互作用することを利用して次の衝突が発生する時刻まで粒子を等速直線運動で動かしてしまうことができる。この方法を Event-Driven と呼び、この方法を用いる MD を Event-Driven Molecular Dynamics(EDMD) と呼ぶ。

TSDMD では時間刻みの間にポテンシャルから受ける力を適当な数値的な積分方法を用いて計算するため、数学的に複雑な方法が用いられその方法によって系の振る舞いが影響を受けたり、また場合によっては解が発散することがあるが、Event-Driven では計算機の数値誤差を除いては系の振る舞いは決定論的であり具体的にどのプログラム、アルゴリズムを用いるかに依存しない。ただし計算効率は用いるアルゴリズムに大きく依存し、計算の困難さはほとんどが次に起こる衝突を効率よく検索するという部分に集中する。本研究のプログラムは村上氏 [6] や礒部氏の方法 [7] を参考に作成した。以下に今回用いたアルゴリズムと境界条件の取り扱いを述べる。以下の議論では慣習に従って粒子直径を 1、粒子質量を 1 とする単位系をもちい、密度については面積あたりに何個の粒子があるかという面積あたり密度の表記を用いる[1]。

2.1　Event-Driven 型 MD の特徴

TSDMD では粒子に連続的なポテンシャルが設定されている。例えば 1987 年の Rapaport の論文 [1] では

$$V(\mathbf{r}) = \begin{cases} r^{-12} - r^{-6} + \frac{1}{4} & (r < r_c = 2^{1/6}) \\ 0 & (r \geq r_c) \end{cases} \tag{2.1}$$

[1] 最密充填の密度を 1 とする単位系を使用している論文もある

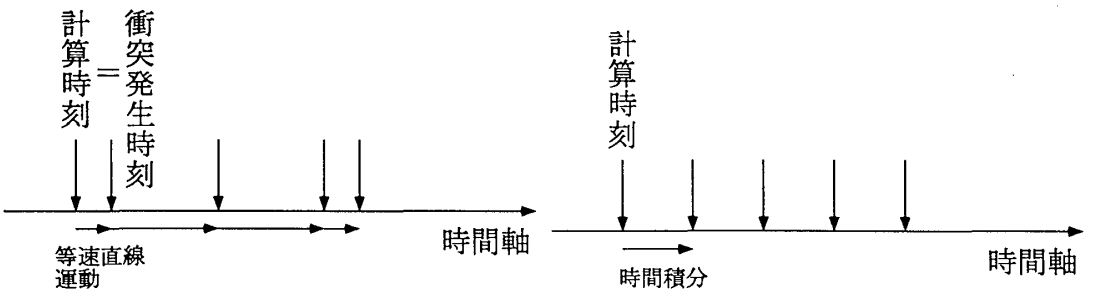

図 2.1: TSDMD と EDMD の違い (左が EDMD、右は TSDMD)、矢印の点が計算される時刻を表す

という形のポテンシャルが設定されている。ポテンシャルがわかればそのポテンシャルから粒子にかかる力を計算することができ、運動方程式を立てることができる。すなわち

$$m\frac{\mathrm{d}\mathbf{x}_i}{\mathrm{d}t} = -\nabla(\sum_{j\neq i} V_j(\mathbf{r})) \tag{2.2}$$

($\mathbf{x}_i$ は i 番の粒子の位置、V_j は j 番の粒子の作るポテンシャル)
粒子はこの運動方程式に従って運動するので、あとはこの運動方程式を数値的に解けばよい。解法としては例えば 4 次 4 段のルンゲクッタ法が挙げられる。このポテンシャルは時間だけでなく空間にも依存するので予測子修正子法を用いるとさらに精度を上げられる。一般に数値解法は系のエネルギー保存を保たないので、この解法には十分な精度を有するものを使用しないとエネルギーの発散などが起こることがある。

粒子の作るポテンシャルの形はある程度の距離で打ち切るようにし、遠くの粒子同士は相互作用しないようにするのが一般的である。これはある程度以上の距離の粒子を考慮に入れないで運動方程式を計算するためである。もしすべての粒子対が相互作用するとするとその計算量は $N(N-1)$ なので $O(N^2)$ であるが、近距離の粒子同士のみ相互作用するとした場合の計算量は平均的に近距離にいる粒子の数を C として CN であり、$O(N)$ で済むからである。

EDMD では相互作用は衝突時のみ瞬間的に行われる、ポテンシャルで書けば

$$\phi(r) = \begin{cases} \infty & (r < R) \\ 0 & (r > R) \end{cases} \tag{2.3}$$

に相当する。物理的にこういうポテンシャルをもつ粒子は存在しないが、近づくと反発しある距離以下にはなかなか近づかないという粒子の典型的な性質を簡単に表現しているモデルである。このポテンシャルを持つ粒子の運動を計算するには、衝突が発生するまでは相互作用がないので等速直線運動で粒子を動かしてしまい、しかるのち瞬間的に行われる相互作用の結果を計算し、また次の衝突まで等速直線運動するという処理を繰り返すことになる。相互作用を完全剛体として近似して取り扱うことにより衝突の処理が瞬間的に行われることになり、厳密に取り扱うことができる。つまりシミュレーションの際には現実の系の連続性を離散化する必要があるが、TSDMD では計算時に時間の離散化を行って計算するのに対し EDMD では粒子モデルの時点ですでに離散化されているという違いである。計算の際の時間進行の様子は図 2.1 に示した。

現象を再現するためには TSDMD と EDMD のどちらが効率がよいかという比較は明らかではないが、1992 年の Rapaport の論文 [8] では、10^5 オーダーの粒子数の計算で完全剛体の EDMD のほうが効率がよいという結果を得ている。

2.2　粒子のモデルおよび計算式

粒子は完全剛体円盤としているので、粒子間の相互作用は衝突時のみに瞬間的に行われる。衝突後の速度は運動量保存則とエネルギー保存則から求めることができる。粒子の位置、速度をそれぞれ $\mathbf{r}_i$、$\mathbf{v}_i$、質量を m_i とすると、衝突後の速度は

$$\mathbf{v}_i' = \mathbf{v}_i - \frac{2\mathbf{r}_{ij}\cdot\mathbf{v}_{ij}}{\mathbf{r}_{ij}^2}\frac{m_j}{m_i+m_j}\mathbf{r}_{ij} \tag{2.4}$$

$$\mathbf{v}_j' = \mathbf{v}_j + \frac{2\mathbf{r}_{ij}\cdot\mathbf{v}_{ij}}{\mathbf{r}_{ij}^2}\frac{m_i}{m_i+m_j}\mathbf{r}_{ij} \tag{2.5}$$

$$\mathbf{r}_{ij} = \mathbf{r}_i - \mathbf{r}_j \tag{2.6}$$

$$\mathbf{v}_{ij} = \mathbf{v}_i - \mathbf{v}_j \tag{2.7}$$

である。第二項が運動量の交換を表現しており、粒子の質量以外は共通なので記憶しておくとプログラム上で再利用することができる。

また Event-Driven の計算を行うためには粒子対の衝突予定時刻が必要である。粒子対が衝突するまでの時間は次の式で与えられる

$$t_{\mathrm{collision}} = \frac{-\mathbf{r}_{ij}\cdot\mathbf{v}_{ij} - \sqrt{|\mathbf{r}_{ij}\cdot\mathbf{v}_{ij}|^2 - \mathbf{v}_{ij}^2(\mathbf{r}_{ij}^2 - (\mathrm{R}_i+\mathrm{R}_j)^2)}}{\mathbf{v}_{ij}^2} \tag{2.8}$$

($\mathrm{R}_i, \mathrm{R}_j$ はそれぞれの粒子半径)
なお $\mathbf{r}_{ij}\cdot\mathbf{v}_{ij} > 0$ の場合は粒子が遠ざかっていく相対速度の場合に、式 (2.8) の根号の中が負になる場合は衝突しない相対速度を意味するので衝突しないものとして処理してよい。

2.3　アルゴリズム

最も基本的で単純な Event-Driven 型の MD シミュレーションは次の手続きを繰り返すことで時間進行を行う

1. すべての粒子の組み合わせに対し、衝突予定時刻を計算する
2. すべての粒子の組み合わせの中から、衝突予定時刻の最も早いものを探しそれを次の衝突時刻とする
3. 2 で求めた次の衝突時刻まで、粒子を等速直線運動させる
4. 衝突を処理する (完全弾性衝突など)
5. 1 に戻る

この手続きに従って次々に衝突を処理しながら時間進行していくことで粒子系の運動を追跡することができる。しかし、このアルゴリズムでは 1 回の衝突を処理するたびに 1 における衝突予定時刻の計算が粒子数 N に対して $N(N-1)/2$ 回行われることになり、系が大規模になるにつれて計算時間は $O(N^2)$ で大きくなってしまうため大規模計算向きではない。

そこでまず容易に考えられる改良としては、衝突した粒子は速度が変化するため衝突予定時刻が変化するが、衝突しなかった粒子同士の衝突予定時刻は変化しないことを利用しその計算結果を再利用することである。つまりある 2 粒子の衝突を処理して 1 に戻ったとき、すべての組み合

わせに対して衝突予定時刻を計算するのではなく衝突した2粒子と他のすべての粒子の組み合わせについて衝突予定時刻を再計算しその他の組み合わせについては前回の結果をそのまま利用することができる。これによって1回の衝突につき計算時間は$2N$程度になり大幅に改善する。

しかしある程度の密度を持った粒子系であれば、遠く離れた粒子同士がその後ずっと等速直線運動して衝突する可能性はきわめて低い。このことを利用し、衝突予定時刻の計算を互いに近くにいる粒子に限定すると大幅に計算量を削減することができる。どの粒子が近くにいる粒子かわかっていれば衝突時に再計算する衝突予定時刻は衝突した粒子の近傍だけでよく、その計算コストは平均的に近くにいる粒子の数に比例し、全体の粒子数によらないので$O(1)$である[2]。また近くにいる粒子についてのみ衝突予定時刻を記録するのでメモリの節約にもなる。このような原理に基づき計算量を削減する方法はいくつか提案されており、今回はその中から最も効率が良いとの研究報告 [7] のある Extended Exclusive Particle Grid Method & Neighbor List の方法を基本に、プログラムの作成改良が行いやすいように修正して使用した。以下にこの方法と、その他使用した高速化技法を述べる。

2.3.1 格子によって近くの粒子を探す方法

近い将来に衝突する可能性が高い近い距離にいる粒子対を発見するためには粒子間の距離を計算しなければならないが、これを全粒子対について計算していては$O(N^2)$の計算になってしまう。そこで空間を格子に区切り、粒子がどの格子に所属するかという情報を利用して近い距離にいる粒子対を絞り込む方法が一般的である。格子サイズの決め方には、取り扱い方にはいくつかバリエーションがあるが、ここではEPGM(Exclusive Particle Grid Method)[3]を使用した。EPGMでは格子の一辺の長さl_gを

$$R < l_g < \sqrt{2}R \tag{2.9}$$

(Rは粒子半径)

を満たすように設定し、各粒子について自分の所属する格子を中心に5×5の格子に所属する粒子を近接した粒子とみなす。格子サイズの下限は中央の格子に所属する粒子が確実に5×5の範囲内に収まる条件であり、上限は一つの格子に2個の粒子が所属することがないようにするための条件である。高密度領域ではこれで十分であるが、低密度領域では5×5の領域に粒子がほとんど見つからず、非効率的になることがある。そこで礒部氏はこれを拡張し5×5に限らずより大きな範囲を見るようにし、さらにその範囲を円形を近似するように設定するEEPGMを用いている。またEEPGMの格子と粒子の対応関係をつけた時点で各粒子について近接粒子の登録を完了したものとみなし、再計算の際には格子をスキャンすることで近接粒子を探している。

今回作成したプログラムでは、EEPGMをもとにプログラムを書きやすくするため少々改造を加えたものを利用している。検索範囲としては7×7の領域を検索し、発見された粒子について再度距離の計算を行って近接粒子を絞り込んだうえで各粒子の持つNeighbor Listに記録するようにした[4]。EEPGMの格子と粒子の対応関係をつけた時点で止めず、近接粒子の記録配列を持たせるようにしたのは各粒子に関する情報をそれぞれの粒子単位で持たせることでデータ構造のわかりやすさを追求したかったためと、近接粒子の検索処理とそのあとの毎回の衝突ごとの再計算処理を完全に分離し、問題ごとに別の仕様のプログラムを組み合わせて使用したり独立に改良していけるようにするためである。また7×7の領域内であってもNeighbor Listに記録する時点で再度

[2] この処理は$O(1)$だが他に少なくとも$O(\log N)$の部分が残るため全体の計算効率は$O(1)$にはならない。しかしこの部分は計算回数は他の部分に比べて圧倒的に大きいため高速化の寄与は大きい

[3] EPGMの利用はもともとTSDMD用に提案されたもので、元は論文 [9][10] にある

[4] 実際に計算するときにはそれぞれの粒子を基点に7×7を検索するのでなく、それぞれの粒子から右側の3×7と下の3マスを検索するようにし、発見した粒子は相互に登録するようにすると検索時間を短縮できる

距離の計算を行って規定以上の距離の粒子を排除しているのは、衝突ごとの再計算の処理時間を低減するためである。なお本プログラムではこの距離計算の際に 2 次元空間上での距離による方法でなく、xy 方向それぞれについての座標の差で行っているので、Neighbor List に登録される粒子はある粒子を中心とする正方形の範囲にある粒子になる。この方法を利用した理由は x 方向と y 方向の速度がともに 1 であった場合、円形の検索では速度 $\sqrt{2}$ で取り扱う最大速度を 1 で扱うためにこの Neighbor List の有効期限が 1.41 倍になり、円形の検索に比べて検索領域が $4/\pi = 1.27$ 倍になる効果を差し引いても有利であると考えたためである。ただし、円形の検索を行った場合との性能比較は行っていない。

2.3.2　近接粒子表の有効期限の設定

2.3.1 小節の方法を用いて作られた Neighbor List は作られてからしばらくの間は衝突する可能性のある粒子をすべて示しているが、一定距離にある粒子までしかリストしていないためある程度時間が経過するとリストにない粒子と衝突する可能性が出てくる。この Neighbor List の有効期限を求める方法が Dynamical Upper Time Cut-off(DUTC) である。一般的な円の検索範囲の場合には、まず全粒子から最大の速度を持つ粒子を探し出す。発見された最大速度を $v_{\max}$ と書くことにすると、有効期限は

$$\tau = \frac{R_{\mathrm{search}} - 2R}{v_{\max}} \tag{2.10}$$

(τ は有効期限、R_{search} が検索範囲、R が粒子半径)
である。同様に正方形領域の検索範囲の場合は、$v_{\max}$ として x 方向速度の最大値 (絶対値で) と y 方向速度の最大値のうち大きいほうを採用する。τ の表式は式 (2.10) と同じであるが、R_{search} としては正方形の一辺の半分を用いる。

最大速度が有効期限まで一定であれば有効期限まで有効でよいが、場合によっては衝突や境界条件での加速によって最大速度 ($v_{\max}$) が上昇することがある。そのため、最大速度を変化させる可能性があるイベントが発生した後には必ず変化した粒子の速度を $v_{\max}$ と比較し、最大速度が更新された場合は

$$\tau_{\mathrm{new}} = \frac{v_{\mathrm{oldmax}}}{v_{\mathrm{newmax}}}\tau_{\mathrm{old}} \tag{2.11}$$

として有効期限を短縮する必要がある。

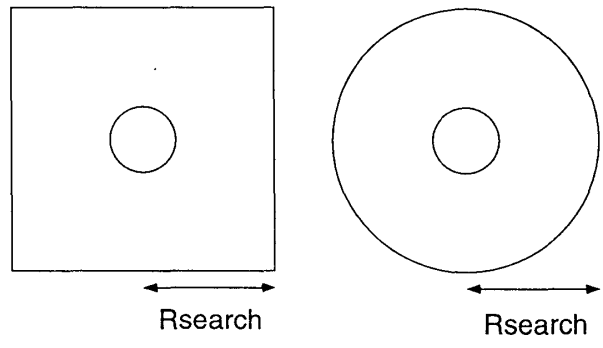

図 2.2: 検索範囲と R の定義

2.3.3　衝突予定の保持方法について

各粒子については Neighbor List に従って衝突予定時刻の計算を行うが、衝突が起こるたびに全粒子が Neighbor List に従った再計算を行ったのでは非効率的である。そこで計算した衝突予定時刻を記録しておき、衝突に関与しなかった粒子の衝突予定時刻を再利用することにする。今回はこの方法として Single-Event Time List を使用した。Single-Event Time List では各粒子についてそれぞれ次に衝突する予定の相手と時刻を最も早い時刻のもののみ記録しておく方法である。衝突が発生した際には衝突に関与した粒子とその周りの粒子についてのみ衝突予定時刻の再計算を行えばよく、他の大多数の粒子についてはそのままでよい。この方法のメリットは結果として得

られるイベントリストの大きさが粒子数と同じサイズに固定されており、その中から最も早いイベントを検索することが容易なことにもある。

他の方法としては衝突予定時刻を計算する際にNeighbor Listに存在するすべての粒子対について衝突予定時刻を記録しておきそのなかからもっとも早い衝突を検索するMultiple-Event Time Listと、すべての粒子対について衝突予定時刻を記録しておくものの検索は各粒子についてもっとも早い衝突予定時刻について行うLocal Minima Algotithmなどが知られている。この中ではLocal Minima Algorithmが最も効率がよいとする研究結果[11]が報告されているが、今回は簡単のためとメモリ容量を節約するためにSingle-Event Time Listを用いた。

2.3.4 完全二分木による次の衝突検索

Single-Event Time ListやLocal Minima Algorithmを用いた場合、粒子の数だけ衝突予定時刻を並べたリストから最も早い衝突を探すことになるが、この方法としては完全二分木(Complete Binary Tree)が便利である。この方法は1対1の勝負を繰り返して優勝を決めるトーナメント方式と原理的に同じである。まず粒子数Nに対して$2N-1$個の配列を用意しN番から$2N-1$番までに粒子番号(ないしポインタ)を書き込んでおく。次に配列の後ろ2つ($2N-1$と$2N-2$)に入っている粒子の衝突予定時刻を比較し、早いほうを勝者として$N-1$番の配列に番号を記録する、以後順に$2M-1$番と$2M-2$番の比較を行って勝者を$M-1$番に記録する処理をMをNから1ずつ減らしながら行うと、最後に配列1番に最も早い衝突予定時刻を持った粒子が来る。この方法の利点はトーナメント表のうち変化しなかった部分を再利用できることで、Neighbor Listを構築した直後はこの方法で全部についてトーナメントを行って最も早いものを検索するが、衝突処理を終えたあと、次の最も早い衝突予定時刻を検索する際には衝突予定時刻が変化した粒子についてのみトーナメントをやり直せばよい。しかも衝突予定時刻が更新された粒子がトーナメントに負け、それより上が変化しない場合(勝者が以前のトーナメントと同じになった場合)はそこでトーナメントの再検索を打ち切ってよい。これによって再検索の計算量が大幅に削減される。

2.3.5 等速直線運動の省略による高速化

全系の中で最も早い衝突が発見された後はその時刻まで粒子を等速直線運動させるのであるが、この計算量はNに比例するため結構大きい計算量になる。これを削減するため、衝突をおこした2粒子とその近くにいる粒子(つまり衝突した粒子のNeighbor Listに登録されている粒子)については等速直線運動で時間を進めるが、他の粒子はそのままにしておくDelayed State Algorithm[11]を使用した。これを実現するため各粒子は現在どの時刻にいるのかという情報を記録している。衝突した2粒子とともに近くにいる粒子の時間を進めるのは、衝突予定時刻を計算するときには互いに同じ時刻で無いと計算しづらいためである。またNeighbor Listの再構築を行う際には全粒子の時刻を最新の時間に合わせる必要がある。

2.3.6 アルゴリズムまとめ

以上をまとめて、次のような手続きで計算を行った

1. 全粒子の時刻を合わせる
2. 各粒子がEEPGM(の簡略版)のどの格子に所属するかを計算して求める。結果は格子と粒子の双方に記録する

3. 全粒子について自分の所属する格子の周り 7×7 個の格子を検索し、発見された粒子のうち所定の距離範囲にいる粒子を粒子の持つ Neighbor List に記録する。EEPGM はこの処理が終われば破棄してよい
4. DUTC 法によって Neighbor List の有効期限を求める
5. 全粒子について、それぞれの粒子の持つ Neighbor List にある粒子との間の衝突予定時刻を計算し、もっとも早い衝突をもたらす相手の番号と衝突予定時刻を記録する。(Single Event Time List)
6. 全粒子の衝突予定時刻について CBT のトーナメントを行う
7. CBT の結果、トップにいる粒子の衝突予定時刻を次の衝突予定時刻とする
8. 7 によって求められた次の衝突予定時刻が DUTC によって求められた有効期限を超えていれば、DUTC の有効期限まで粒子を等速直線運動させた上で Neighbor List と SETL の情報を破棄し、1 にもどる。
9. 発見された次の衝突予定時刻まで、衝突する 2 粒子と衝突する 2 粒子の Neighbor List にある粒子を等速直線運動によって時間進行する。
10. 衝突による速度変化を処理する
11. 衝突した粒子と、衝突した粒子の Neighbor List に登録されていた粒子について、衝突予定時刻を再計算する。
12. 衝突予定時刻の変化した粒子について CBT によるトーナメントをやりなおす
13. 7 にもどる

また、各粒子は次のような情報を保持している

1. 位置、速度
2. 固有時刻
3. 近傍にいる粒子の数と番号のリスト
4. 次の衝突予定時刻とその相手
5. (必要ならば) 粒子半径、質量

2.4　境界条件

計算領域の端は周期境界か、熱浴として処理した。周期境界は境界に到達した瞬間に粒子が移動するものとして処理し、熱浴は粒子が熱浴に触れた瞬間にそれまでもっていた速度を失い新しく熱浴の設定温度と設定速度に従う分布を持った速度をランダムに与えるものとする。これは熱浴の境界から粒子が出て行った瞬間に同じ場所から新しい粒子が進入してくるものと考えることもできる。原理的には必ずしも出て行った場所から進入してくる必要は無くまた一時的に粒子数が増減するモデルもありうるのだが、もし熱浴からの粒子数の増減を許したとすると系に粒子を押し込むときに粒子を押し込める隙間を探さなければならないためプログラム上の実現が難しい。

周期境界であっても流れのある方向の周期境界の際には、一種の熱浴のような処理を行った。周期境界に到達した粒子は周期的に移動させるのであるが、その際にそれまでの速度に関わり無く新しい速度を与えてしまうという方法である。これによって周期境界の端で流速を一定にすることができ、連続体近似の非圧縮流体シミュレーションでよく用いられる速度固定の流入境界条件を再現することができる。

以下にそれぞれの場合の具体的な扱いを述べる。

2.4.1 熱浴

熱浴に衝突した粒子は今まで持っていた速度を忘れ、所定の温度に相当する速度を持つように速度を再設定されるが、この際、熱浴の壁に沿う方向と垂直な方向では扱いが異なる。以下では熱浴の壁が $y=0$ に存在するものとして話を進める。温度 T の系にある粒子は一般に、ある方向 (例えば x 方向) について速度 v である確率を ψ とすると

$$\psi(v) = (2\pi T)^{-\frac{1}{2}} \exp(-\frac{v^2}{2T}) \tag{2.12}$$

であらわされるガウス分布である。よって熱浴の壁と平行な x 成分については速度をこれで与えればよい。しかし、熱浴表面から系に進入する粒子の y 方向速度分布は、$v_y < 0$ の粒子はそもそも進入してくることはなく、y 方向に大きな速度を持った粒子は高い確率で系に進入してくるためこれとは異なる。この影響を考慮するためには、$v_y < 0$ の分布確率を 0 にし、$v_y > 0$ には速度に比例する重率をかけてやればよい。この重率 w を式で表現すると

$$w = \begin{cases} 0 & v < 0 \\ v & v \geq 0 \end{cases} \tag{2.13}$$

である。熱浴から進入する粒子はこの重率をガウス分布にかけて規格化した分布になるので

$$\phi(v) = \begin{cases} 0 & v < 0 \\ \frac{1}{T} v \exp(-\frac{v^2}{2T}) & v \geq 0 \end{cases} \tag{2.14}$$

これで xy 方向ともに速度の確率分布が求まったので、乱数にしたがって速度を与えることができる。区間 [0:1] の一様乱数 r からこの乱数を得るためには逆関数法と呼ばれる方法を用いる。一様乱数 r から v を得る変換を $v = f(r)$ とすると、この変換が満たすべき関係式は、ある乱数値 r の値は、確率分布の 0 から $f(r)$ までの確率分布の積分値に等しいという関係から

$$r = \int_0^{f(r)} \phi(v)\mathrm{d}v \tag{2.15}$$

$$= \exp\left(\frac{-(f(r))^2}{2T}\right) + 1 \tag{2.16}$$

$$\tag{2.17}$$

これを $f(r)$ について解くと

$$f(r) = \sqrt{2T\log(1-r)} \tag{2.18}$$

また、一様分布の性質より、

$$f(r) = \sqrt{2T\log(r)} \tag{2.19}$$

としても分布は同じであるので、これを用いた。

2.4.2　流速つき熱浴

2.4.1 節では静止した熱浴から進入する粒子の速度分布を求めたが、流速を設定された熱浴から進入する粒子の速度分布はやや複雑である。流速は V_x, V_y が設定されているものと仮定すると、x 方向 (熱浴の壁と平衡方向) について粒子の熱運動の速度分布は

$$\psi(v_x) = (2\pi T)^{-\frac{1}{2}} \exp\left(-\frac{(v_x - V_x)^2}{2T}\right) \tag{2.20}$$

よって x 方向にはガウス分布を V_x だけずらした分布を与えればよい。y 方向については 2.4.1 節と同様に考えると、速度分布

$$\phi(v_y) = (2\pi T)^{-\frac{1}{2}} \exp\left(-\frac{(v_y - V_y)^2}{2T}\right) \tag{2.21}$$

に対し、重率 w をかけて規格化した分布になるので、規格化定数を C として

$$\phi(v_y) = \begin{cases} 0 & v_y < 0 \\ C v_y \exp(-\frac{(v_y - V_y)^2}{2T}) & v_y \geq 0 \end{cases} \tag{2.22}$$

$$\begin{aligned} C &= \int_0^\infty (2\pi T)^{-\frac{1}{2}} v_y \exp\left(-\frac{(v_y - V_y)^2}{2T}\right) \\ &= \frac{1}{\sqrt{2\pi T}}\left(T\exp\left(-\frac{{V_y}^2}{2T}\right) + \sqrt{2T}V_y \mathrm{Erfc}\left(-\frac{V_y}{\sqrt{2T}}\right)\right) \end{aligned} \tag{2.23}$$

$$\text{ただし } \mathrm{Erfc}(x) = \int_x^\infty e^{-t^2}\mathrm{d}t \qquad \text{(ガウスの誤差関数)}$$

ここで通常の熱浴から進入する粒子と同じように逆関数法を用いて一様乱数からこの分布を得ようとすると、積分の結果が式 (2.23) と同様にガウス関数と誤差関数の和の形になってしまうため $f(r)$ について解くことができない。そのため、この分布関数を区間を区切って数値的に積分を行い、その結果を用いて乱数の発生を行った。

具体的には区間ごとに分布関数の値を求めて配列に累積値を記録する、つまり

$$List[n+1] = List[n] + \frac{n\mathrm{d}t}{T}\exp(-\frac{(n\mathrm{d}t - V_y)^2}{2T})$$

というように記憶する。そして十分大きな n まで計算した所で数値積分が完了したとみなし、その合計 sum を記録する。その後一様乱数を発生しそれに sum をかけると区間 $[0 : sum]$ の乱数 r が得られるので、r と配列の分布関数の累積値を比較して累積値が r になる区間を探し、線形補間で v_y の値を求めた。

この計算は計算時間を大きく使うので乱数発生の結果は 1024 個を保存しておき、それをランダムな順番で使いまわすことで計算のコストを節約した。リストからの選択には整数乱数の下位 10bit を使用している。

2.4.3　物体表面

障害物として用いる円柱など流体粒子と比べて規模の異なる大きさの物体表面に接触した流体粒子の取り扱いについては、Rapaport が論文 [1] で挙げているようにいくつか方法が考えられる。

一番単純なのはRapaportが論文で採用しているように物体表面で弾性衝突をするとするものである[5]。しかしこの方法ではエネルギー保存は保たれるが物体表面で摩擦がないことになり、円柱などの障害物にかかる摩擦抵抗が存在しないことになってしまう。今回の研究では流れから物体が受ける抗力なども測定するつもりであるためこの方法は適当でないと判断した。

物体表面で摩擦があるようにする条件は滑りなし条件と呼ばれるが、それを粒子系で実現する散乱方法はエネルギー保存を守るかどうかで大きく2種類考えられる。滑りなし条件を実現するためには衝突後の粒子の速度分布に物体表面の接線方向の速度を持たせてはならない。そのためエネルギー保存を守る場合には速度の大きさを保存しつつ、散乱角度をランダムに選ぶことになる。またはエネルギー保存を放棄し、物体に設定された温度に従った速度分布 (速度の絶対値の分布) を持つように散乱する方法も考えられる。Rapaportは物体がエネルギーの吸収または供給源になるという理由で一定の温度の速度分布を与える方法を嫌っているが、この両方法の違いは物体から流体への熱伝導があるかないかの違いだけであり、温度が十分高く熱速度が系の平均速度に比べて十分大きい極限では結果に違いをもたらさないと推測される。しかし、さほど高くない有限の温度でシミュレーションを行った場合には影響が出る恐れもある。今回行ったシミュレーションではこの危険を避けるためエネルギーを保存する角度散乱を選択した。

方法	エネルギー保存	滑りなし／あり
弾性衝突	する	あり
角度散乱	する	なし
温度固定 (熱浴扱い)	しない	なし

表 2.1: 物体表面の散乱条件

2.5 外力の処理

問題によっては粒子に駆動力として外力を用いている。もし外力を厳密に処理しようとすると一番簡単な一定外力の場合であっても粒子の軌跡は放物線になり、衝突予定時刻は放物線同士の距離が粒子半径の和に等しくなる時刻として求めねばならない。同じ加速度の外力に引かれる粒子同士であれば、速度変化が相殺するため衝突予定時刻は変化しない。だが境界条件との衝突時刻は放物線の軌道を計算しなければならず、また粒子にかかる外力加速度が一様でない場合にはやはり計算が複雑になる。衝突予定時刻の計算はMDシミュレーションの中で最も回数の多い処理であるので、この計算が複雑になることは著しい効率の低下を招いてしまう。そこで外力についてはTSDMDの考え方を導入し、ある一定の時間間隔ごとにその時間間隔の間に受ける速度変化をまとめて与えるいわゆるオイラー法を用いることにした。この速度更新処理を行うと衝突予定時刻が変化してしまうためすべての粒子について再計算の必要がある、そのため外力による速度更新処理を行う際は粒子の時刻合わせを行ってEEPGMによるNeighbor Listの作成から再計算を行うことにした。

2.6 性能

今回作成したシミュレーションプログラムの性能は、Alpha21264A 750MHzの計算機で表2.2のような計算速度が得られた。測定は密度0.6から0.8程度で行っている。アルゴリズムの参考に

[5]このRapaportの論文はTSDMDなので弾性衝突といってもポテンシャルを仮定した衝突である

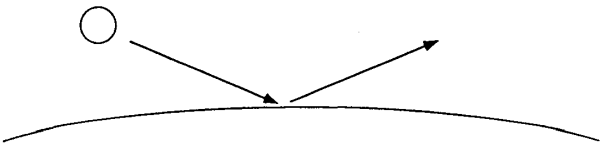

図 2.3: 弾性衝突

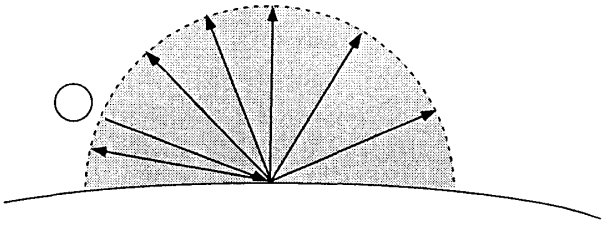

図 2.4: 角度散乱、速度は保ったまま角度がランダムに反射される

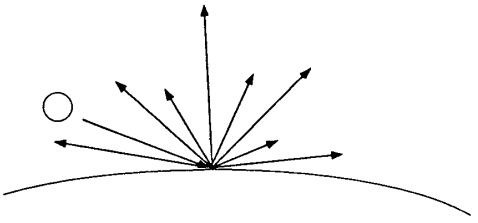

図 2.5: 温度固定、乱数によって反射後の速度の大きさは異なる

した礒部氏の論文に掲載されている速度は Alpha 600MHz で 1.3×10^5collision/second というものである。プロセッサの性能がそのクロック周波数に比例すると考え、今回作成したプログラムで礒部氏と同じ計算機で計算を行ったとすると

$$83000 \times \frac{600}{750} = 66400 \text{ Collision/second}$$

ぐらいの性能になるものと推定される。礒部氏のものに比べおおむね半分の速度に低下しているが、これはアルゴリズムを簡略化した分と、比較的低密度 (0.3 から 0.5) 向きに設計した効果によるものと思われる。

粒子数	衝突処理数/秒
17 万	28000
1 万	54000
2500	83000

表 2.2: 性能表

またTSDMD とどちらが計算性能として有利かを調べるため、渡辺氏による TSDMD 型 MD シミュレーション [12] と性能を比較してみた。計算条件は表 2.3 にまとめてある。計算方法が異な

り、TSDMD では衝突回数が容易に測定されないため、衝突回数での性能の比較はできない。そのためこの場合は計算時間あたりに求められた系のシミュレーション上での時間経過の長さで比較するのが適当であろうと思われる。結果は今回作成した Event-Driven 型 MD シミュレーションのほうが 5 倍以上高速であり、粒子を完全剛体球として扱ってよい問題については Event-Driven が効率のよい解法であるといえるだろう。なお時間あたり衝突処理数が表 2.2 よりも大きくなっているのは高密度のためであると考えられる。

項目	Event-Driven MD	Time-Step-Driven MD
粒子数	23256	23256
密度 (面積あたり粒子数)	1.04	1.04
粒子の平均熱速度	5.06	5.06
衝突回数	26448646	-
シミュレーション上での経過時間	10	180
計算所要時間 (sec)	342	36000
シミュレーション上の時間/計算時間	0.029	0.005

表 2.3: EDMD と TSDMD の性能比較

17 万粒子の場合の計算時間の内訳を表 2.4 に示した。衝突のたびに $O(\log N)$ の計算が必要となる CBT による最短衝突時間の検索が単独で最も計算時間を消費しており、つぎが衝突予定時刻の計算、等速直線運動の計算である。オーダーを計算してみればわかるとおりこれらはみな衝突のたびに衝突に関与した粒子とその近傍にいる粒子すべてに対して必要となる計算であり、そういった部分を改善することが今後の改良に必要である。

計算の流れ別に見ると EEPGM による Neighbor List の作成とそれに伴う全粒子の衝突予定時刻の再計算処理が占める割合は 15%、粒子が衝突するたびに必要となる粒子の衝突処理や周囲の粒子の衝突予定時刻の再計算が占める割合は 73%である。

処理名	計算時間
CBT によるトーナメント	30%
衝突予定時刻の計算	26%
等速直線運動	15%
Neighbor List の作成	9%

表 2.4: 計算時間の内訳

第3章　剛体球系の粘性率測定

流体のシミュレーションによって特定の流れパターンを作るためには、系の粘性率を知ることが必要である。ここでは、剛体球系の粘性率を平行平板間流れを利用して測定した結果を示す。剛体球系の物性を決定付ける量は密度 ρ、温度 T のみであり、粘性率はこの二つの関数として

$$\nu = \nu(\rho, T)$$

として表されるはずである。そのため粘性率の粒子密度と温度に対する依存性をまず議論する。また、熱速度に比べて無視できない大きさの平均流速をもつようになった場合は粘性率等が変化してしまう恐れがあるため、熱速度と平均流速の比率に対する依存性も調べた。最後に系のサイズ依存性についても検討した。以下の議論で用いる粘性率νは一般に動粘性率と呼ばれる量であり、ずり粘性率μとの関係は

$$\nu = \frac{\mu}{\rho}$$

(ρは密度) である。

3.1　モデル

流体の粘性率を測定するためには外力と速度の関係が必要となるため、重力を駆動力として流体を動かし、平行平板流の流速が平衡に達したときの流速を利用して粘性率を求めた。設定された状況は図3.1のような状況で、左右の壁には周期境界条件、上下の壁には熱浴を設定してある。粒子は重力によって右向きの速度を得るが、上下の熱浴と衝突した際に右向きの平均速度を失うことになるので、平衡状態ではy方向について放物型の速度分布を持つことが期待され、実際にそのような平衡状態に落ち着く。

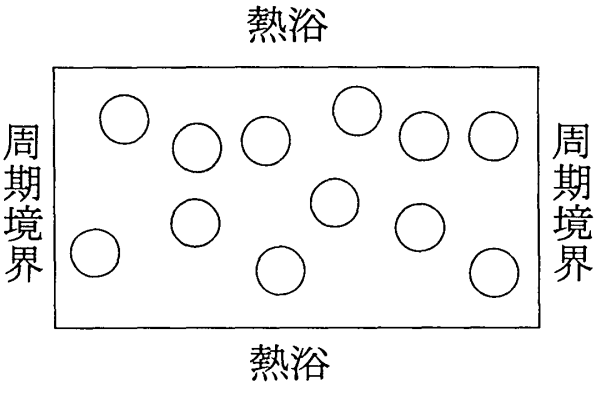

図 3.1: Re測定方法モデル図。全体に右向きの重力をかける

3.2　測定原理

結果は平衡状態において全粒子の平均速
度を測定した。全体が放物型の速度分布を持っていると仮定すると平均流速から粘性率を求めることができる。非圧縮のナビエストークス方程式は一般に

$$\frac{\partial \mathbf{v}}{\partial t} + (\mathbf{v}\cdot\nabla)\mathbf{v} = -\nabla(\frac{p}{\rho}) + \nu\nabla^2\mathbf{v} + f$$

(ρは密度、νは動粘性率、fは外力 (加速度))
外力として加えられた重力の方向をxとし、それに直交する方向をyとする。いま平行平板流を考えているので、x方向の一様性とy方向速度がないことを仮定することにより

$$\nu\Delta v = -f \tag{3.1}$$

$$\nu\frac{\partial^2 v_x}{\partial y^2} = -g \tag{3.2}$$

(gは重力加速度)
とおける。系のy方向長さをL_yと置くと速度分布は

$$v = \frac{1}{2}\frac{\partial^2 v_x}{\partial y^2} y(L_y - y) \tag{3.3}$$

$$= -\frac{1}{2}\frac{g}{\nu} y(L_y - y) \tag{3.4}$$

であるから、平均流速との関係から

$$v_{\text{average}} = \frac{1}{L_y}\int_0^{L_y} v dy \tag{3.5}$$

$$= \frac{gL_y^2}{12\nu} \tag{3.6}$$

$$\nu = \frac{gL_y^2}{12v_{\text{average}}} \tag{3.7}$$

となり、粘性率を求めることができる。

また、無次元化されたナビエストークス方程式は

$$\frac{\partial v}{\partial t} + (v\cdot\nabla)v = -\nabla p + \frac{1}{\mathrm{Re}}\nabla^2\mathbf{v} \tag{3.8}$$

$$\mathrm{Re} = \frac{UL}{\nu} \tag{3.9}$$

(U は流れの速度スケール、L は長さスケール)
である。この式を利用して粘性率から系のレイノルズ数が計算できる。

3.3 結果

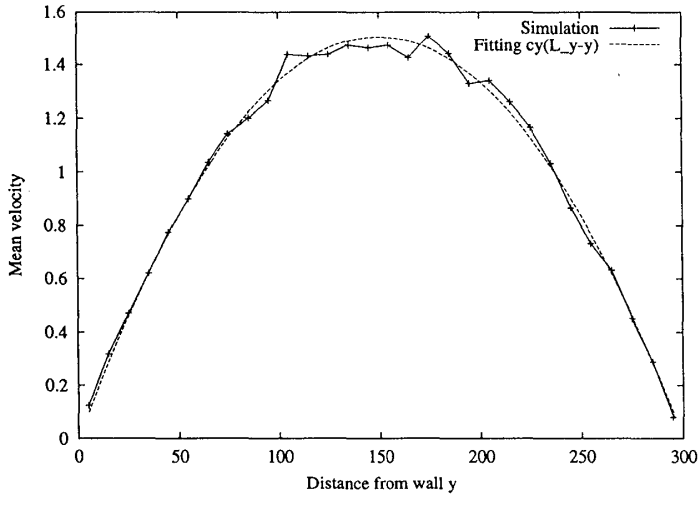

図 3.2: 粘性率測定、速度分布例 ($L_y = 300$)

密度依存性、温度依存性、熱速度と平均流速の比率依存性のそれぞれについて検討を行った。シミュレーションは $L_y = 100$、密度 0.5(粒子/面積)、温度 2 を中心に行った。結果として得られる速度分布の例を図 3.2 に示した。近似曲線は

$$f(x) = cy(L_y - y)$$

の放物線である。この例は $L_x = 300, L_y = 300$ の計算条件で t=10000 のスナップショットである。この場合は系が大きいため熱雑音が数で圧縮されきれいに放物線が見えているが $L_x = 100, L_y = 100$

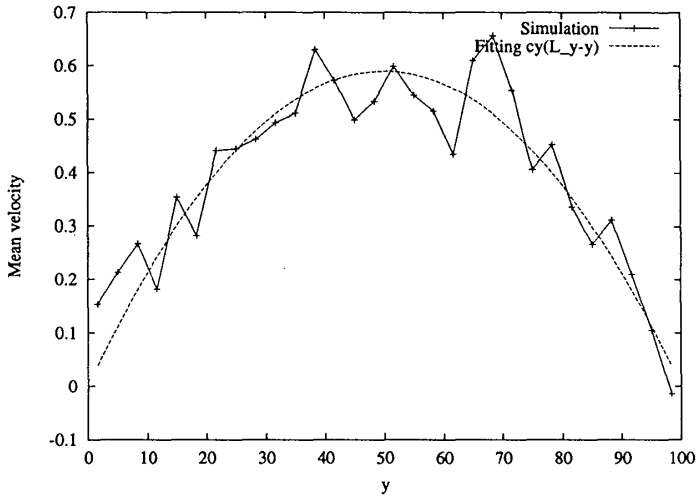

図 3.3: 粘性率測定、速度分布例 ($L_y = 100$)

の場合には図 3.3 のようにもっと荒れたグラフになる。しかしこの流れは定常であり時間平均を取ることで熱雑音を抑えることができるので平均流速の測定への影響は少ない。

平均速度の時間変化は $L_y = 100$、密度 0.5、温度 2 という典型的な場合で図 3.4 のようになる。同時にプロットした近似曲線は $t = 4000$ 以降で定数を仮定してフィッティングを行ったもので、ここから平均流速は 0.42 と判断できる。

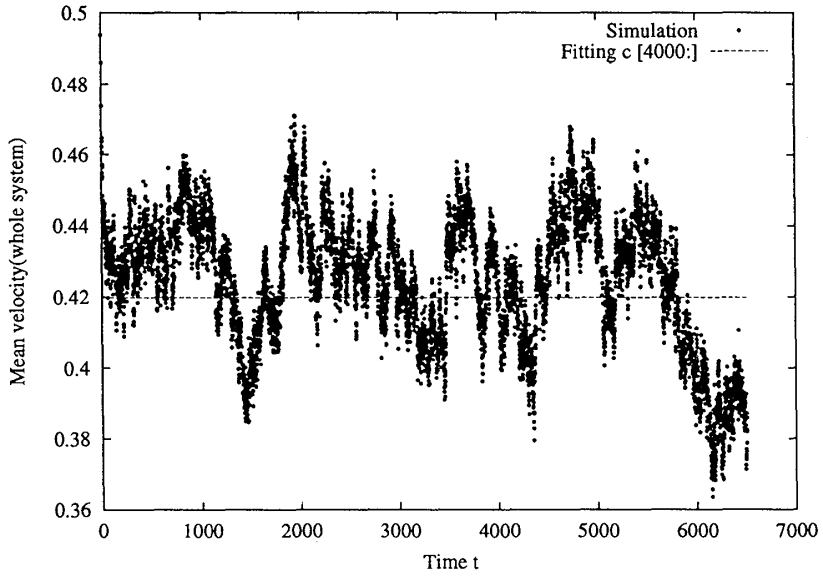

図 3.4: 系全体の平均速度の時間変化 ($L_y = 100$)

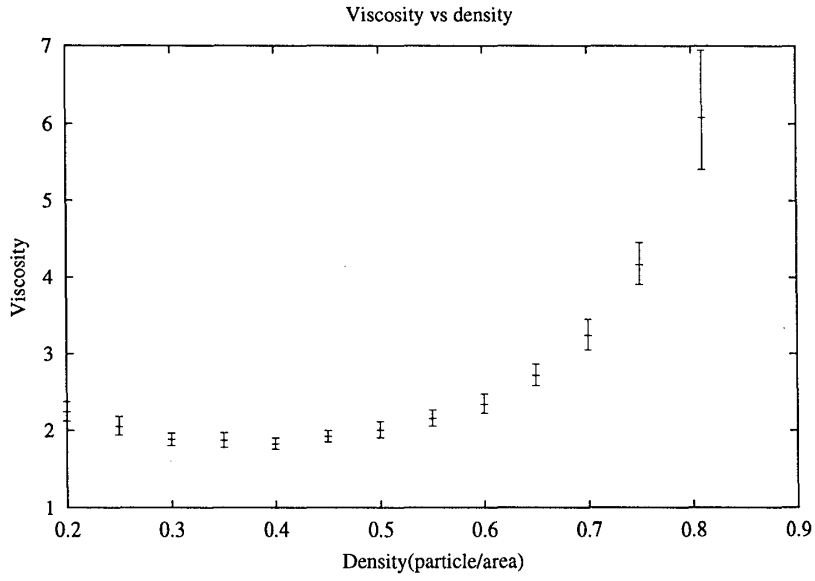

図 3.5: 粘性率の密度依存性

3.3.1 密度依存性

密度依存性についての評価は $L_y = 100$、温度 2 のもとで粒子密度を変えて定常に達したときの流速を測定した。結果は図 3.5 のようになる。誤差は平均速度の揺らぎから標準偏差で求めた。この結果から密度 0.3 から 0.5 ぐらいで動粘性率が最低になるという事がわかる。定性的には高密度側では衝突が激しく性質も固体に近づいていくため動粘性率が上がっていくというのは順当な結果であり、低密度側では動粘性率の定義 $\nu = \mu/\rho$ に含まれる密度の効果が効いているものと考えられる。

3.3.2 温度依存性

温度依存性はシミュレーションではなく原理的に求めることができる。剛体粒子系は時間スケールを変えても全粒子の運動速度と外力のスケールが変わるだけで、粒子運動のパターンは一定である。ここで時間スケールを c 倍にしたとすると、粘性率の計算に用いたそれぞれのパラメータは次のようになる

$$v' = cv \tag{3.10}$$

$$v'_{\text{average}} = cv_{\text{average}} \tag{3.11}$$

$$g' = c^2 g \tag{3.12}$$

よって、T と ν は次のように変化する

$$T' = \frac{1}{2}mv'^2 = \frac{1}{2}m(cv)^2 = c^2 T \tag{3.13}$$

$$\nu' = \frac{gL_y^2}{12v'_{\text{average}}} = c\nu \tag{3.14}$$

よって、T が c^2 倍になると ν は c 倍になる。つまり

$$\nu \propto \sqrt{T}$$

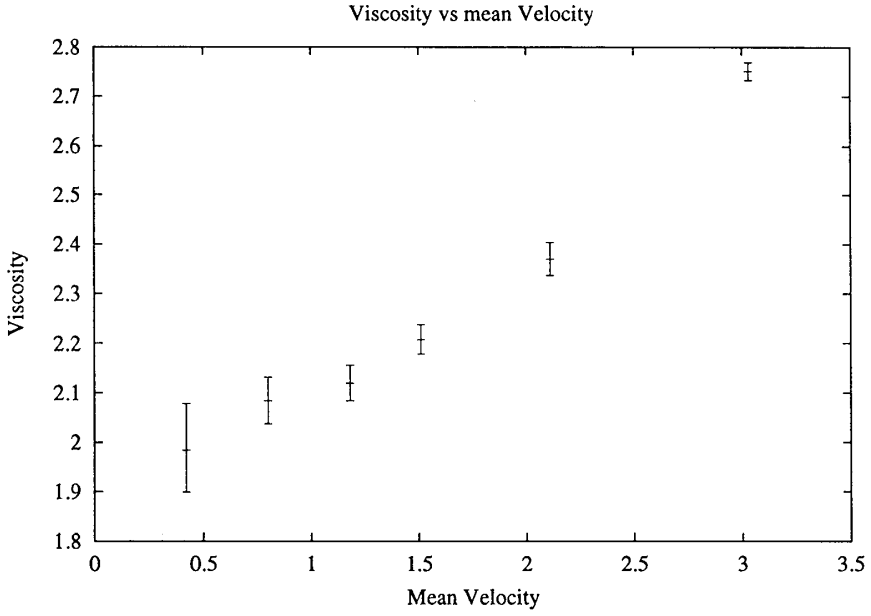

図 3.6: 粘性率の平均速度依存性

この結果を確認するため温度を変えてシミュレーションを行った。結果は表 3.1 のようになった。この式が正しいことを示しているといえるだろう。なお誤差は温度 2 では 5%、温度 8 で 30%程度である。

温度	粘性率	粘性率／ $\sqrt{温度}$
2	2.00	1.42
4	2.80	1.42
8	3.93	1.39

表 3.1: 粘性率の温度依存性

3.3.3　平均流速依存性

粒子が平均的に持つ熱速度に比べ巨視的に見た流速が無視できない大きさになった場合、巨視的な流速が実効的な温度を変化させることで粘性などに影響する可能性が考えられる。基本的に巨視的にみた流速が十分小さい場合には影響はなくなるはずなので、熱速度に比べどのぐらいの大きさまでならば問題なく計算できるかという観点から評価した。結果は図 3.6 のようになった。熱速度の 1/2 にあたる平均速度 1 あたりで 5%程度のずれが発生しており、平均速度 1.5 あたりから急激に上昇している。この結果から、全体でほぼ一様の粘性率を保つためには速度差は熱速度に比べて大きくないことが望ましく、熱速度の 1/2 を超えない程度に設定すべきであると考える。

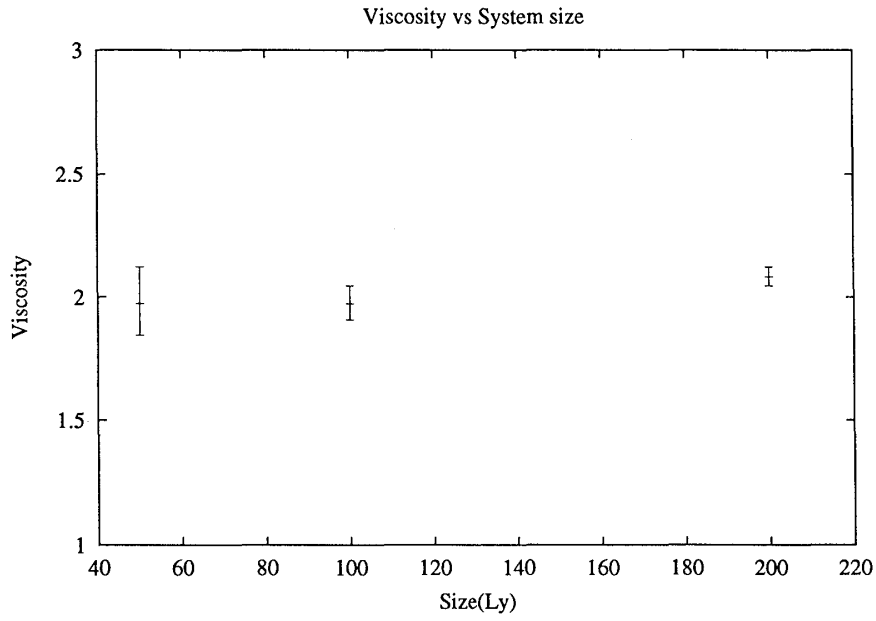

図 3.7: 粘性率のサイズ依存性

3.3.4 系のサイズ依存性

Wainwright らの論文 [13] などで指摘されているように、2 次元の剛体粒子系には物性定数に系のサイズ依存性がある恐れがある。そのため系のサイズを変えて粘性率を測り、サイズ依存性を調べた。その結果を図 3.7 に示す。結果的にこのサイズ範囲で粘性率にサイズ依存性はみられなかった。もっと大きな系では影響が見られるかもしれないが、当面計算に用いるサイズ (100-1000 程度) では影響は無いようである。このことから、今回測定した粘性率の値は、少なくとも今回計算に用いるスケール範囲で一定とみなしてよいと考えられる。

3.3.5 まとめ

以上のような性質と測定結果より、粘性率の密度依存性は図 3.5 で得られ、温度については $T^{-1/2}$ に比例することがわかった。また系の平均運動速度が熱速度に達する程度まで加速するとその発熱が無視できなくなるため、熱速度の半分以下ぐらいに抑えるべきであることがわかった。

この結果から、系を所定のレイノルズ数に設定するためには

- 高レイノルズ数の場合は、0.3 から 0.5 あたりが粘性率が低く高レイノルズ数を再現しやすい。粘性率最低を示しており、多少の密度変化に対して粘性率が変化しにくい 0.4 を使用するか、粒子数を低減して計算コストを下げるため 0.3 を採用するのが適当であろう。
- 低レイノルズ数の場合は、高密度を使用するのが粘性率上昇のためによいがあまり高密度にして Alder 転移点に近づくと粘性率の密度依存性が激しくなり系全体の粘性率を一定にできなくなる恐れがある。そのため密度は 0.8 程度に抑えるのが適当であろう。

第4章　円柱を過ぎる流れ

典型的な流体シミュレーションである円柱を過ぎる流れにおいて、古典的な流体シミュレーション法であるSMAC法で得られた結果とMDで得られた結果を比較した。なおSMAC法については文献[14]などにより、流体は非圧縮を仮定している。計算はすべて2次元であり遠方で一様流を仮定している。

4.1　方法

4.1.1　計算領域

計算に用いた設定状況は図4.1に示したようなものである。シミュレーションプログラム上で取り扱いが難しい粒子数の変化をなくすため、流入境界と流出境界は周期境界で結ばれている。また流出境界から流入境界へと周期境界で移動した粒子は、その瞬間にそれまで持っていた速度を忘れ、新たに系の設定温度と設定流入速度で決まる速度付き熱浴の速度分布を持つ速度を与えられる。通常は流入口から流出口への周期境界越えの移動もできるようにしてあるが、周期境界を過ぎる流量を一定にするため流量が不足している場合にはこの移動を禁止している。この流量調整処理を行わない場合、計算領域の幅と円柱の大きさによって流量の不足が生じるが、この調整処理を行うことで計算領域の幅 L_y が円柱直径の4倍を超えるぐらいの条件では所定の流量が確保できるようになる。それ以上細い流路を使う場合は流れに対する抵抗が大きいので他の方法が必要であろう。系の上下の端は流速つきの熱浴となっており、上下端に達した粒子は右向きの平均流速と系の設定温度に従った速度分布をもつ速度を与えられる。

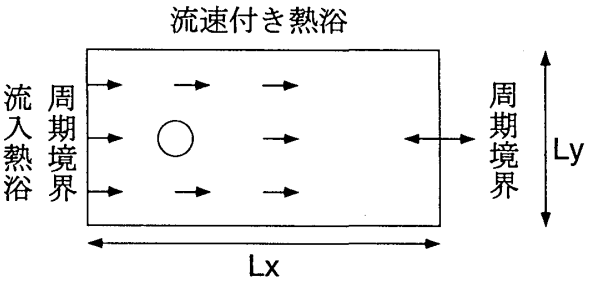

図 4.1: 計算領域の設定

円柱表面にはエネルギーを保存するランダム角度散乱条件を用いた。円柱に衝突した粒子はその速度の絶対値は変えずにランダムな角度に散乱されることになる。格子を用いるシミュレーションでよく用いられるNo-Slip条件では物体表面で流体の接線方向速度を0としている。MDにおいては角度散乱によってこの状況を再現することができる。

4.1.2　密度分布

流体は非圧縮を想定してSMACシミュレーションや実験値との比較を行っているが、MDで行った場合には設定温度が十分に高くないためある程度の密度変化が生じる。しかしその密度変化は流速を熱速度の半分以下に抑えているためさほど大きくなく、円柱後方で平均密度の10%程度の低下である。図4.2はレイノルズ数Re=110の設定でシミュレーションを行ったときの密度分布で、後方の渦発生領域で10%程度の密度低下が起こっていることがわかる。なお流速が小さい設定の場合はこの密度低下はこれよりも小さくなる。

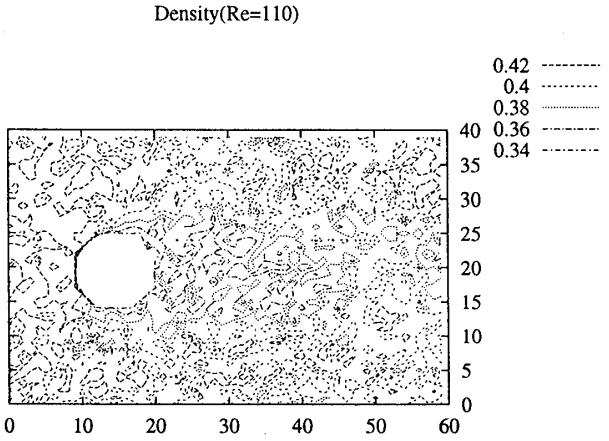

図 4.2: Re=110 の流れにおける粒子密度の等高線プロット。円柱後方の低密度領域でおおむね10%程度の密度低下が発生している。

4.1.3 可視化

速度場の可視化は計算領域を長方形のブロックに区切り、その中に存在する粒子の合計運動量を平均密度とブロック面積で割って行っている。

$$\text{表示速度} = \frac{1}{\text{平均密度} \times \text{ブロック面積}} \sum_{\text{ブロック内}} \text{粒子速度}$$

熱速度に対して流速を半分以下に抑えるように設定してあるため、流れ場を書く際には熱速度を平均化することでランダムな熱速度を抑圧する必要がある。ランダムな熱速度を抑えるためには粒子速度の平均化を行うサンプル数を増やさなければならず、サンプル数に対して熱によるノイズは

$$S \sim \frac{\overline{v_{\text{thermal}}}}{\sqrt{N}} \tag{4.1}$$

(S はノイズの振幅、N はサンプル数、$\overline{v_{\text{thermal}}}$ は平均熱速度)
という関係で減少するので、少なくとも流速 v_{flow} に対して $v_{\text{flow}} > S$ というようにしないと流速を見ることができない。平均化のためのサンプル数を増やす方法には

1. ブロックサイズを大きくとる
2. 粒子密度を上げる
3. 時刻を変えてサンプルを取る

というような方法が考えられるが、高レイノルズ数側と低レイノルズ数側で適切な方法が異なる。方法 1 のブロックサイズを大きくとる方法は、あまり大きくとると流れを表現できなくなってしまうため、ブロックサイズの一辺の大きさは障害物の大きさの数分の一が限界である。障害物を大きく取れる高レイノルズ数側ではこれが有効であるが、低レイノルズ数側では効果がない。流速を 1/2 に落とせば同じレイノルズ数で障害物を大きく取れるが、その分ノイズを 1/4 に落とさなければならず平均流速とノイズの比率は向上しない[1]。

[1]なお 3 次元系であれば、流速を 1/2 にすると障害物直径とブロックサイズの比が一定でもサンプル数を $2^3 = 8$ 倍に取れるので、ノイズは $1/\sqrt{8} = 1/2\sqrt{2}$ となり改善する

方法2の密度を上げる方法では、粒子数の増加に伴って必要計算量が増加するのと、粘性率が上がってしまうので、高レイノルズ数のシミュレーションには向かない。逆に低レイノルズ数の計算にはこの方法が有効で、粘性率が上がるため障害物を大きくとれるので密度上昇分以上にノイズを抑圧できる。ただし密度が固液転移点に近づくと粘性率の密度依存性が激しくなるためあまり転移点近くまで上げるのは危険であろう。

方法3の方法は定常な場合は可能であるが、非定常な場合は適用できない。また完全に相関を忘れるぐらい長いサンプリング間隔でとらなければならないのでシミュレーションの時間が長くなる欠点がある。

まとめると定常な低レイノルズ数側では密度を上げるか時間平均をとってノイズを抑えるのが有効であり。非定常な高レイノルズ数側では系を大きく取るため自然にノイズが圧縮される効果を利用し、足りなければ系の密度を上げるのが適切と考えられる。

4.2　流れパターン

円柱を過ぎる流れの流れパターンをレイノルズ数別に計算した結果を示す。流れパターンには一般に表4.1のような関係が知られている(出典は[15])。また比較の対象としてMAC法の派生計算法の一つであるSMAC法を用いて直交等間隔格子で計算した流れ場を示した。それぞれのレイノルズ数で流れパターンが再現されている事がわかる。

Re=6の流れでは後方に流体が滞留しているような領域ができており、よくみると中に渦ができていることがわかる。

レイノルズ数範囲	流れのパターン
Re<1	前後対称の流れ
5<Re<40	後方に渦発生
40<Re<70	後流が脈動
70<Re<200	カルマン渦発生
200<Re	カルマン渦が乱れる

表4.1: 流れのパターン表

Re=31の流れでは円柱後方に長く伸びた渦が出ている、流速が流入流速に比べて小さいのでわかりにくいが円柱後方で円柱に向かう流れが観測され対の渦が後流中にできていることが確認される。MDシミュレーションの結果ではSMACに比べて後流の伸び方が少し弱い感じになっているが、これは流出境界条件の影響が出てしまっているものと考えられる。SMAC法の境界条件は流入側と流出側にまったく関係がないため境界ぎりぎりまで後流が伸びているが、MDでは周期境界で速度をリセットするもののその影響が流出側にも少し及んでしまい少々早めに後流が消えるようになるものと考えられる。

Re=110では渦が剥離しカルマン渦となる。流れのパターン、渦の間隔ともに両シミュレーション間でよく一致しており流体をよく再現しているといえるだろう。ストローハル数は0.22で、実験値よりやや高い。これも流出境界がやや近かった影響が考えられる。

4.3　抗力測定

円柱にあたる流れから円柱が受ける力を測定した結果を示す。Experimentとしてプロットしたものは文献[16]による実験値で、Simulationとあるものが今回の計算結果、SlipBoundaryとしたものは円柱表面の衝突を単純な弾性反射として計算した参考用のデータである。Simulationとして示した結果は実験値とよく一致しており、正しく流体を再現しているといえるだろう。よい結果を得るためにはおおよそ流路の幅が円柱の5倍以上、円柱の位置が流入境界から円柱の2直径

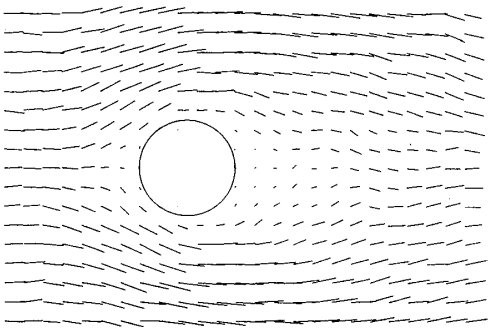

図 4.3: Re=6 の流れ、MD シミュレーション。線は流れの方向と速度を示す

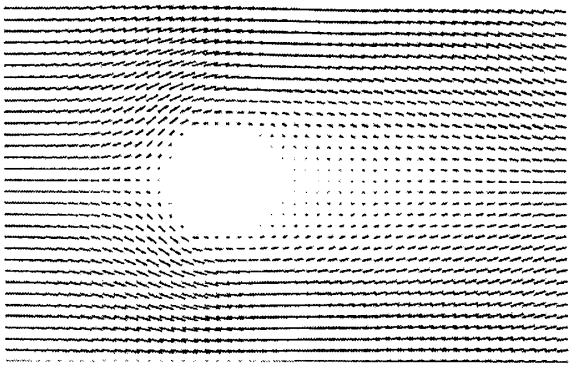

図 4.4: Re=6 の流れ、SMAC シミュレーション。線は流れの方向と速度を示す

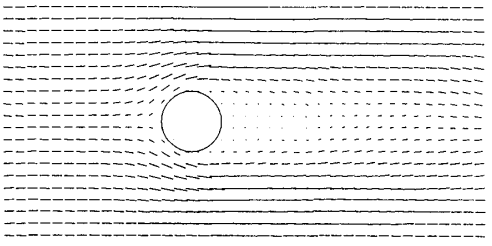

図 4.5: Re=31 の流れ、MD シミュレーション。線は流れの方向と速度を示す

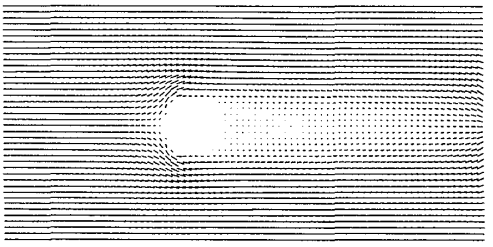

図 4.6: Re=31 の流れ、SMAC シミュレーション。線は流れの方向と速度を示す

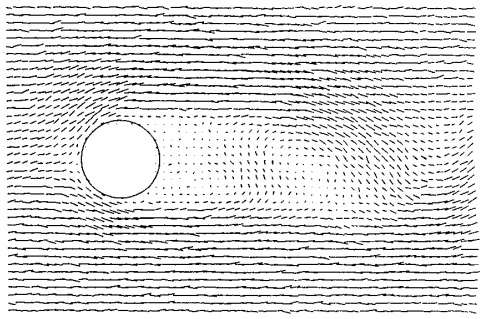

図 4.7: Re=110 の流れ、MD シミュレーション。線は流れの方向と速度を示す。カルマン渦が確認される

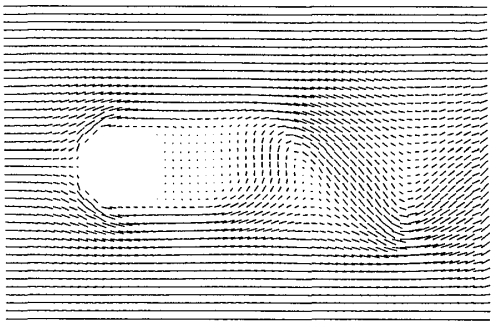

図 4.8: Re=110 の流れ、SMAC シミュレーション。線は流れの方向と速度を示す。カルマン渦が確認される

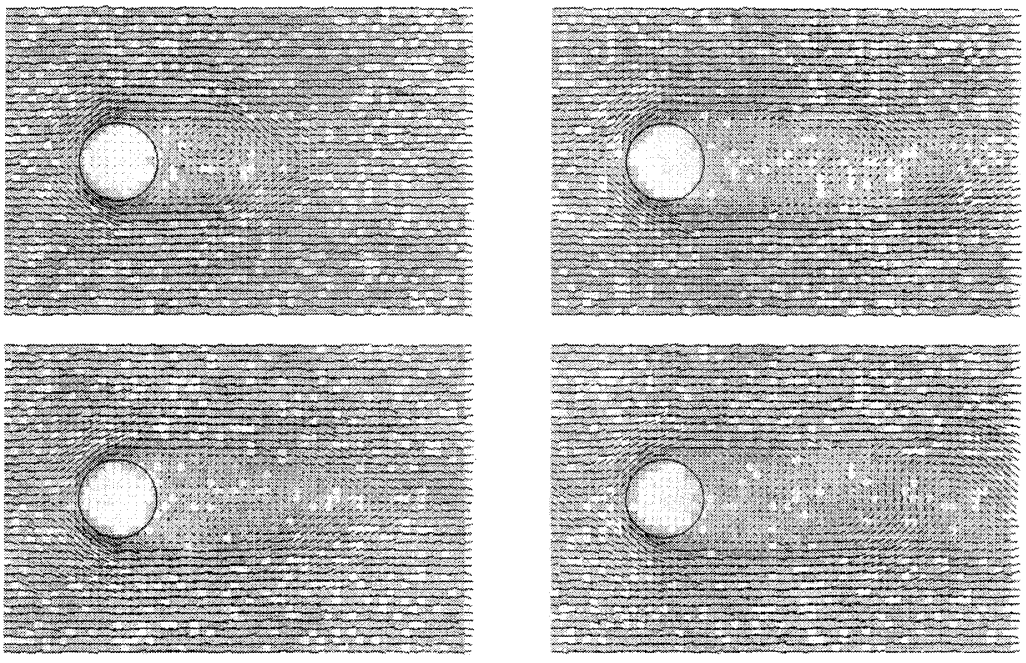

図 4.9: MD シミュレーションによる Re=110 の流れの成長、左上、左下、右上、右下の順に t=500,1000,1500,2000、地についている濃淡は渦度の強さを表現している

以上に設定するのがよい。また計算領域を広く取ったものほど抗力の値が実験値に近づく傾向も見られる。同時に SlipBoundary としてプロットした弾性反射のシミュレーション結果は、滑りあり条件と対応するものであるが、この場合は高レイノルズ数で渦の発生が滑りなし条件の場合に比べて小さく、抗力が小さくなっている。これにより No-Slip の状況を再現するためには角度散乱が適当であることがわかる。

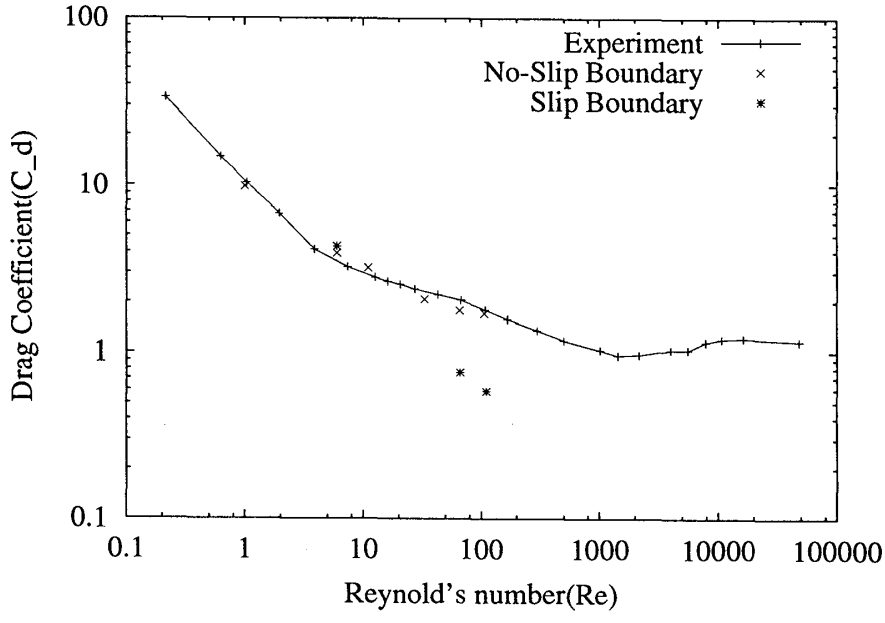

図 4.10: 抗力係数グラフ

第5章　二層流のKelvin-Helmholtz不安定のシミュレーション

MD シミュレーションでは界面の概念が必要なく、混合する流体を容易に扱えるという特徴を生かし、二層流の Kelvin-Helmholtz 不安定のシミュレーションを行った。

5.1　二層流の Kelvin-Helmholtz 不安定

Kelvin-Helmholtz 不安定は速度の異なる流れの界面に発生する不安定のことである。互いに逆方向に向かおうとする流れの界面には渦が生じ、その渦は流れと渦自身によってによって引き伸ばされ崩壊する。

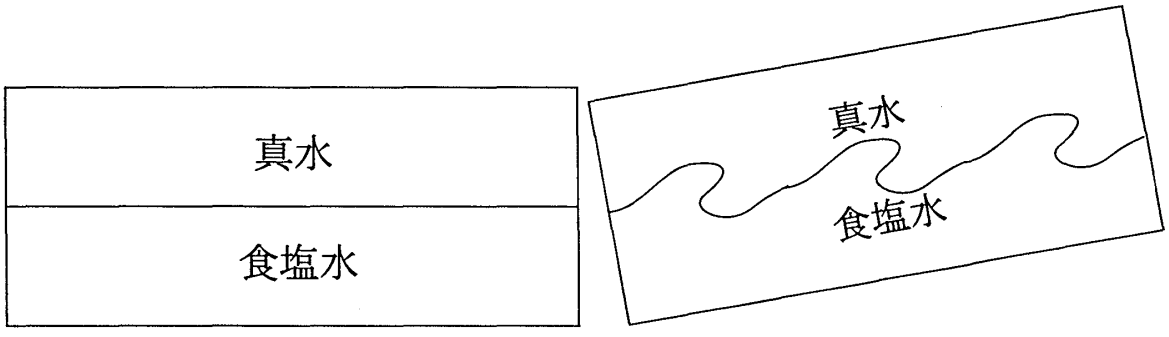

図 5.1: 左図:初期状態。 右図:傾けて波立ち始めた状態。

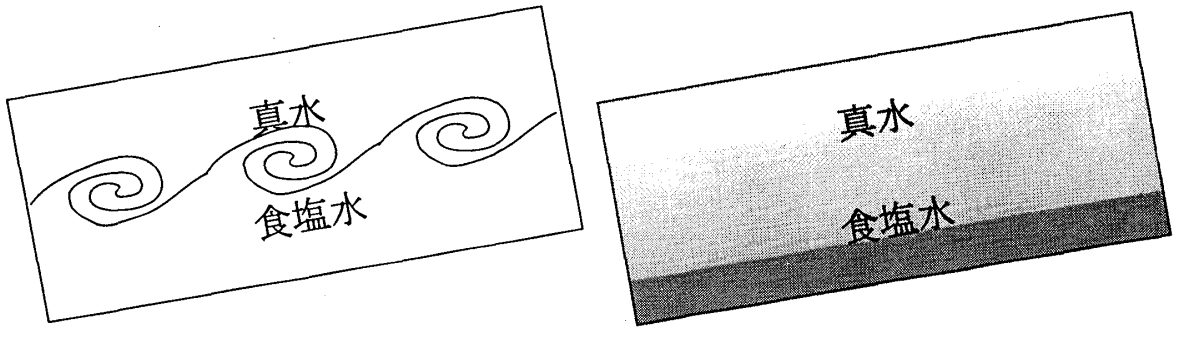

図 5.2: 左図:渦が大きく巻いた状態。 右図:渦が崩壊し濃度勾配が残った状態。

実験的によく行われる方法では [17]、細長いパイプを用意しその中にまず真水を半分入れる。そのあと食塩水を真水と混合しないように静かに入れ、2層の状態にしてパイプを水平にする。そうしてからパイプの端を持ち上げて傾けると、食塩水は比重が重いためパイプの下になったほうへ向かって流れ、真水は逆に上の方向へ流れるので界面に流速差が生じる。すると渦が発生し、成長するとすぐに崩壊、その後は密度勾配を残したまま混合流体が流れていくようになる。

食塩水と真水には界面を見やすくするために色をつけておくのが一般的である。また食塩水と真水の境界は入れた直後には 1cm 程度であるが、その後しばらく置いておくことで拡散させ、境界層の厚さを制御することが可能である。この境界層の厚さによってできる渦の波長や振幅が変化する。

渦成長の過程をよくみると、初期の平らな界面から少し凹凸ができると不安定性からそれが拡大する。ある程度大きな凹凸になると反対の流速を持つ流れに押し流され、図5.1(右) のようにひしゃげた凹凸になる。その後次第に渦になって食塩水と真水の両方の流体を巻き込み図5.2(左) のようになり、接触面積が大きくなることから拡散が促進されて渦が崩壊する。崩壊した後には上下の濃度勾配は残るものの特徴的な界面は失われ図5.2(右) のようになり、そのまま流れていくようになる。

5.2　計算モデル

流速差を再現し混合の様子を見るため灰色と黒の2種類の粒子を用いた。灰色の粒子には左向きに重力がかかっており、黒の粒子は逆に右向きに重力がかかっている。これは食塩水と真水がパイプを傾けたときに比重の違いにより左右に流れようとする力を再現するためである。境界条件は図5.3のように設定した。上下の端は弾性反射のみを行う単純な反射壁で、左右は周期境界である。

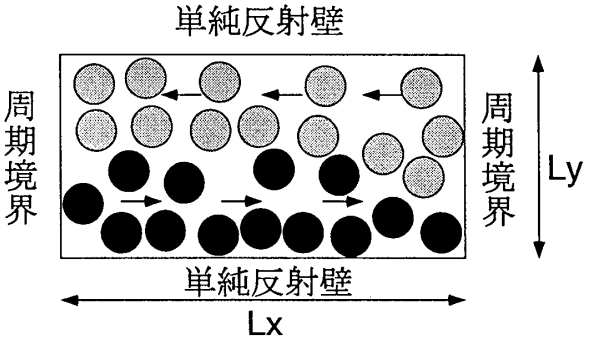

図 5.3: 二層流の Kelvin-Helmholtz 不安定、計算モデル

5.3　結果

結果は図5.4のようになり、渦が発生して崩壊する過程が再現されていることがわかる。流れ場を解いて流線を書くといった方法でなく粒子単位でシミュレーションを行っているため、拡散して渦が崩壊するまで連続的にシミュレーションが可能である。また重力の強さを変えることで拡散速度と渦成長の速度の比が変化し、渦の振幅が変化することも確認している。

第6章　まとめ

今回の結果で剛体円盤 MD で十分に流体は再現できることがわかった。粘性率や壁面の扱いなどについて今回調べたデータを利用することで、他の計算条件のもとでも計算が可能であろう。

円柱などの物体を過ぎる流れで言うと、レイノルズ数が大きくなるほど多くの粒子数が必要であり、粒子数に従って大きな計算力を要するため Re=100 程度の流れまでしか再現できなかった。しかしその結果は十分に一致していたといえるだろう。抗力は実験値とよく一致し、誤差は最大でも 20%程度である。また本シミュレーション方法の利点を生かし、二層流での Kelvin-Helmholtz 不安定による渦発生から崩壊までを容易にシミュレーションできることを示した。

この方法が効果を発揮するのは二層流の Kelvin-Helmholtz 不安定のように界面が変形し、また拡散していくような状況や、砂粒のような物体が多数流体の中に混入しておりそれが境界条件を複雑にしているような場合であると思われる。このような連続体近似によるシミュレーションが困難な状況において、MD による方法のメリットが生かされることがあるだろう。

最後に残された課題を挙げる。

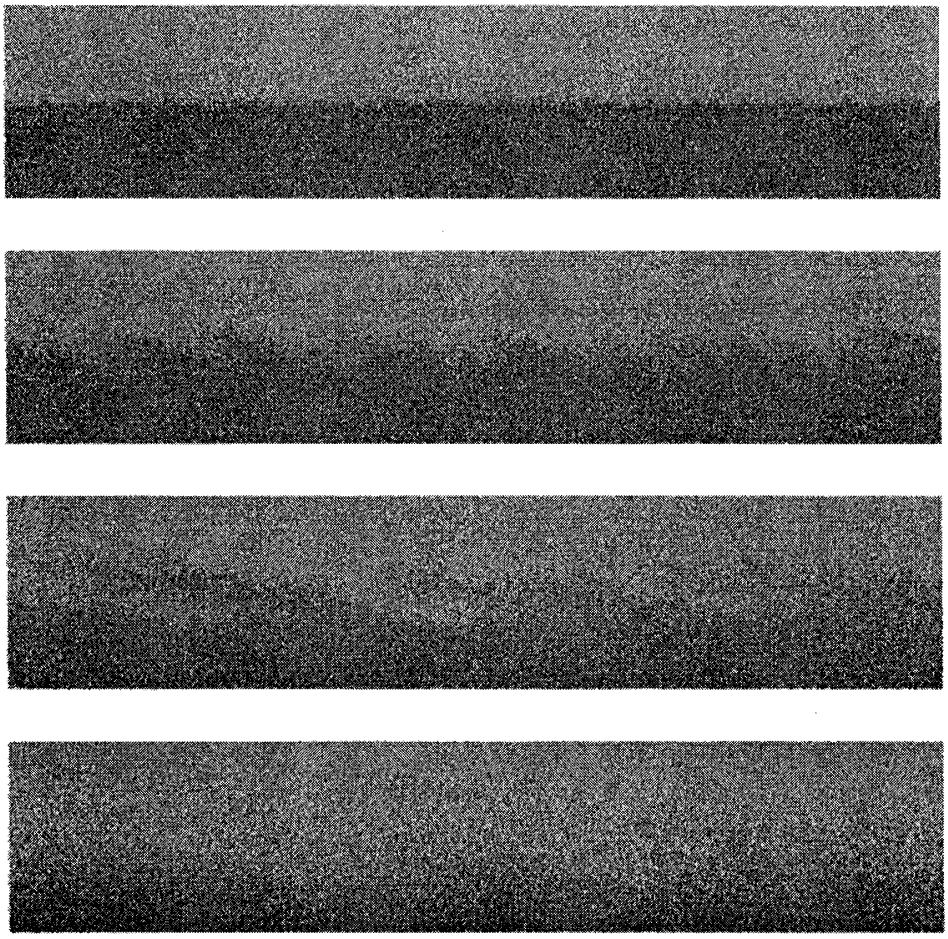

図 5.4: 二層流の Kelvin-Helmholtz 不安定、結果。時刻は上から t=10,200,400,900

- 運動物体の入ったシミュレーションへの応用
  今回行えなかった運動物体の入ったシミュレーションへの応用が考えられる。

- 大規模計算での更なる計算量削減
  乱流の MD シミュレーションを行えるぐらい大規模な計算をするためにはまだまだ規模が足りない。更なる計算効率の向上 (または計算機能力の向上) が必要である。

- 圧縮性流体への応用
  今回は基本的に非圧縮流体か圧縮性が重要でない問題を扱ったが、剛体円盤系は基本的には圧縮性である。圧縮性の程度を制御する方法がわかれば圧縮性流体の計算に利用できるはずで、その方法の開発が必要である。

- 粘性率の操作方法の検討
  粘性率を何らかの方法で操作できるようにすると、粒子数を増やさずに高レイノルズ数のシミュレーションが可能であったり、極低レイノルズ数の計算がやりやすくなるかもしれない。衝突処理の際の反発方向を単純な弾性反発にするのではなく、人工的に操作するなどの方法が考えられる。

第7章　謝辞

本研究の間お世話になった方に、ここに感謝を述べたいと思います。

伊藤伸泰先生には流体のシミュレーションをやるといって以来色々なテーマや方法を提案していただきました、また研究会などに参加させていただきいろいろと考えさせられました。また本研究に直結するところでは高速化の方法やシミュレーションの対象などについてもアイデアを頂きました、ありがとうございました。

助手の湯川諭さんには計算機の使い方から乱数の発生方法までいろいろなことの相談に乗ってもらいました。

研究員の Hans-Georg Matuttis さんには粉体の研究をされている関係で、2次元円盤で山を作る方法について教えていただきました。

博士課程の村上輝好さんには今回のプログラム等で非常にお世話になりました。Event-Driven 型 MD のアルゴリズムと参考資料のほとんどは村上さんに教えていただいたものです。アルゴリズムの実装と高速化の相談には何度も乗っていただき、おかげで速くコードを書き上げることができました。

同じく博士課程の島田尚さんと渡辺宙志さんには、シミュレーション対象の選定の議論などに付き合っていただきました。また修士論文の書き方のようなところも教わりました。

また研究室外では、河村哲也先生には MAC 法のシミュレーション方法を教えていただきました。また流体力学一般や、色々なシミュレーションの例などを紹介していただき、お世話になりました。

皆様ありがとうございました。

付 録A　SandRippleの計算

運動する粒子を含んだシミュレーションの例として、リップルの計算を行った[1]。これは現象としては砂漠などに現れる風紋や、川底や海岸などで見られる砂の波打った模様に現れるものである。通常こういった現象は流れ場を連続体近似の流体シミュレーションで解き、その流れ場の流速に流速と砂輸送量の関係を規定したモデル方程式を適用して砂を移動させる。今回行ったシミュレーションでは砂も粒子単位で扱うことでモデル不要の計算を目標にしている。

今回行った結果では高いところにある粒子が大きく飛ばされたり、粒子が集団で山を形成して移動しているように見えるなどリップル形成においてみられる現象が動画レベルでは見られた。しかし静止画像で見て明確な砂山ができるレベルにはいたらず崩れてしまう。2次元粒子で円盤の山を作るのは難しいという事情も効いているようである。

A.1　モデル

粒子は通常の大きさの粒子の中に直径 4-8 倍の大きさの粒子を混ぜて行った。大きい粒子には下向きに重力をかけており、基本的に底に沈んでゆく。系の上端には流速が設定されており、そこで得た運動量が拡散的に伝達され、大きな粒子に伝わって動くようなモデルである。左右は周期

[1] 明確な砂山とならなかったため付録とした

境界になっており、速度の操作は行っていない。

A.2 結果

結果は図 A.2 や図 A.3 に示したような様子になる。表面から 6 層ぐらいにある大粒子は小粒子の衝突を受けて右に運動しており、速度は上にある大粒子ほど早くなっている。大粒子にはある程度グループ的なものができており、山となって運動しているような様子がみられるが静止画でそれがわかるほど明確ではない。最も上の層にある大粒子は飛ばされて運動していて、他の大粒子と衝突せずに計算領域の 1/3 ぐらいを移動して次のグループに飲み込まれるといった様子がみられる。

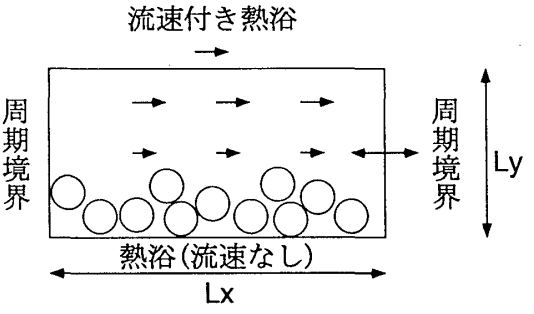

図 A.1: 計算領域の設定

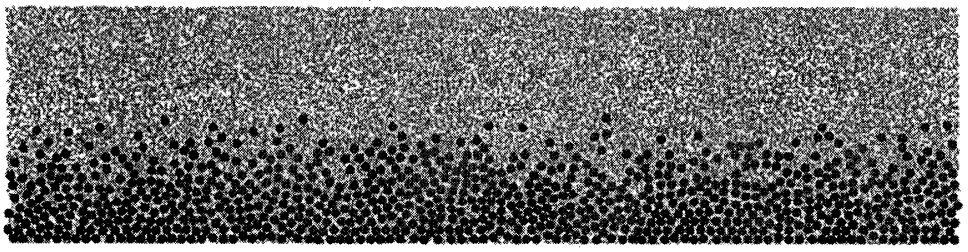

図 A.2: 計算結果のスナップショット、大粒子が砂粒、小粒子が流体粒子を表現

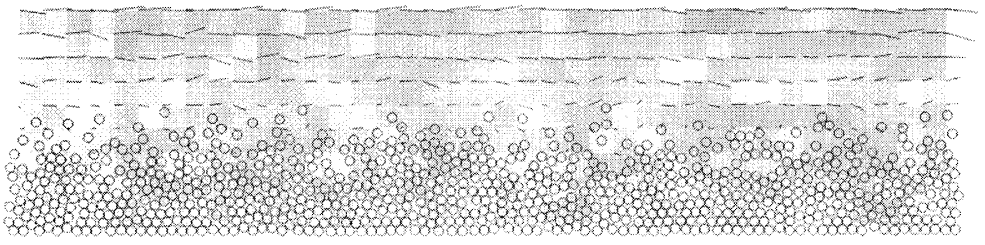

図 A.3: 計算結果のスナップショット、流れ場と大粒子の位置。濃淡は渦度を表現している。

図 A.4 は大粒子の速度分布である。表面の粒子ほど高速で移動していることがわかる。図 A.5 は密度分布であり、上のほうが薄くなっている様子がわかる。もし粒子がほとんど動かないか熱速度でランダムに運動しているようなものであれば速度分布はこのようには出ず、密度分布も鋭く落ちるはずである。なお y=100 付近で密度が大きく変動しているのは粒子サイズの影響である。

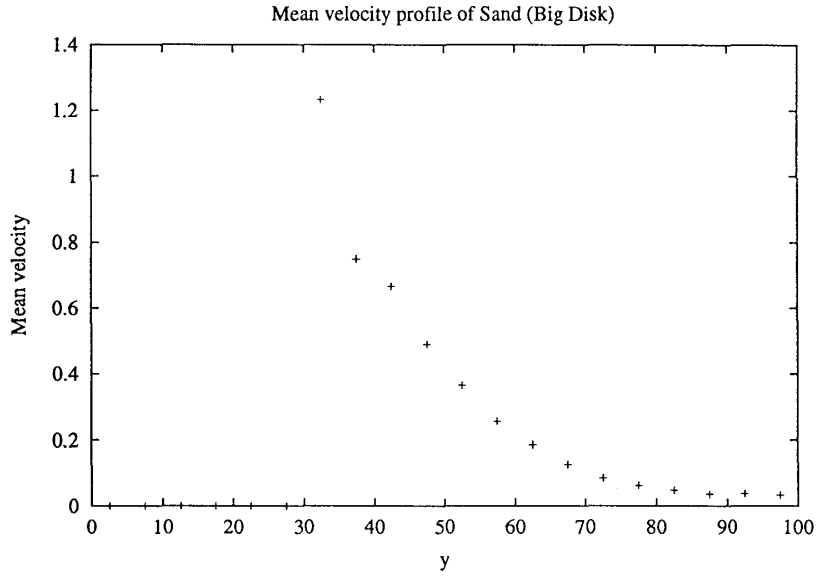

図 A.4: 大粒子の底からの高さ別速度プロファイル

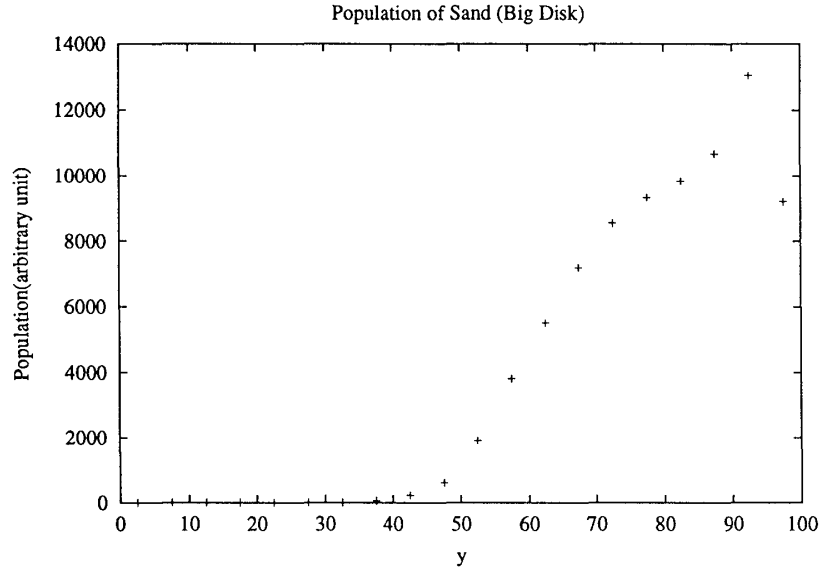

図 A.5: 大粒子の底からの高さ別密度分布

参考文献

[1] D.C. Rapaport, *Microscale hydrodynamics: Discrete-particle simulation of evolving flow patterns*, Phys. Rev. A **36** (1987) 3288.

[2] D.C. Rapaport, *Molecular-Dynamics Study of Rayleigh-Bénard Convection*, Phys. Rev. Lett. **60** (1988) 2480.

[3] A. Puhl,M. Malek Mansour, and M. Mareschal, *Quantitative comparison of molecular dynamics with hydrodynamics in Rayleigh-Bénard convection*, Phys. Rev. A **40** (1989) 1999.

[4] K. H. Andersen, Marie-Line Chabanol, Martin van Hecke, *Dynamical models for sand ripples beneath surface waves*, preprint(cond-mat/0003475/)

[5] 菅牧子, 数値シミュレーションの乾燥地環境問題への応用, お茶の水女子大学大学院人間文化研究科博士論文 (2000)

[6] 村上輝好, 剛体粒子系におけるエネルギー輸送の数値的・統計物理学的研究, 東京大学工学系研究科修士論文 (2000)

[7] Masaharu Isobe, *Simple and efficient algotithm for large scale molecular dynamics simulation in hard disk systems*, Int. J. Mod. Phys. C **10** (1999) 1281.

[8] D.C. Rapaport, *Unpredictable convection in a small box: Molecular-dynamics experiments*, Phys. Rev. A ,**46** (1992), 1971.

[9] V. Buchholtz,T. Poschel, *A vectorized algorithm for molecular-dynamics of short-range interacting particles*, Int. J. Mod. Phys. C **4** (1993), 1049.

[10] W. Form, N. Ito, G. A. Kohring, *Vectorized and parallelized algorithms for multimillion particle MD-simulation*, Int. J. Mod. Phys. C **4** (1993) 1085.

[11] M. Marín, D. Risso, and P. Cordero, *Efficient algorithms for many-body hard particle molecular dynamics*, J. Comput. Phys. **109** (1993) 306.

[12] Hiroshi Watanabe,*Simulational Study on Two-dimensional Melting*, 東京大学工学系研究科修士論文 (2001)

[13] T.E. Wainwright, B.J. Alder, and D.M. Gass, *Decay of Time Correlations in Two Dimensions*, Phys. Rev. A **1** (1970) 18.

[14] 河村哲也, 流体解析 1, 朝倉書店 (1996)

[15] 神部勉, 流体力学, 裳華房 (1995)

[16] 豊倉富太郎, 亀元喬司, 流体力学, 実教出版 (1976)

[17] S. A. Thorpe, *Experiments on the instability of stratified shear flows:miscible fluids*, J. Fluid Mech. **46** (1971) 299.